Bayesian Belief Networks: From Construction to Inference

Bayesiaanse Belief Netwerken: Van Constructie tot Inferentie

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

TER VERKRIJGING VAN DE GRAAD VAN DOCTOR AAN DE UNIVERSITEIT UTRECHT OP GEZAG VAN DE RECTOR MAGNIFICUS, PROF. DR. J.A. VAN GINKEL, INGEVOLGE HET BESLUIT VAN HET COLLEGE VAN DECANEN IN HET OPENBAAR TE VERDEDIGEN OP DINSDAG 13 JUNI 1995 DES OCHTENDS TE 10.30 UUR

door

Remco Ronaldus Bouckaert

geboren op 27 januari 1967
te Haarlem

Promotor: Prof. dr. J. van Leeuwen
Co-promotor: Dr. ir. L.C. van der Gaag
Faculteit Wiskunde en Informatica

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Bouckaert, Remco Ronaldus

Bayesian belief networks : from construction to inference
/ Remco Ronaldus Bouckaert. - Utrecht : Universiteit
Utrecht, Faculteit Wiskunde en Informatica
Thesis University Utrecht. - With summary in Dutch.
ISBN 90-393-0848-9
Subject headings: Bayesian belief networks ; learning /
Bayesian belief networks ; inference.

Acknowledgements

First of all, I want to thank Linda van der Gaag for the tremendous effort she put in making this thesis a more readable document. I would like to thank Jan van Leeuwen, my promotor, for carefully reading the concept manuscripts and his constructive comments. Furthermore, I would like to thank the reading-committee prof. dr. ir. E. Backer, prof. dr. G. Cooper, prof. dr. R. Gill, and prof. dr. J.J. Meyer for reviewing my thesis. Last but not least, I thank Christien Koeleman for the graphical design of this thesis.

Contents

Introduction

Reasoning with uncertainty is more common than reasoning without. Based on just a limited number of observed events we decide to perform an action. However, the events that we observe are in many cases not sufficient to determine in an exact way the consequences of the action we decided to perform. One is painfully aware of this process when voting in governmental elections.

The uncertainty involved in decision-making processes originates from various sources. It may stem from incompleteness of information or arise from imprecise definition of variables. Further, the complexity of relations between variables means that relations often are approximated such that errors are introduced. All these sources of uncertainty are almost continuously present in real-life decision making processes. Humans process uncertain information routinely, and hence appear to be well equipped for handling uncertainty. However, some decisions have far reaching consequences. This motivated the development of formalisms to model uncertainty for the purpose of making optimal decisions.

1.1 Historical Background

The first and most widely used mathematical formalism for dealing with uncertainty is probability theory. As early as the sixteenth century [63] probability theory has been used to describe uncertainty and to help in decision making. The development of probability theory was clearly inspired by decision making with far reaching consequences, namely, gambling. There are two main approaches to probability theory: the frequentist approach and the much later developed subjective Bayesian approach. The frequentists are willing to accept probabilities only if they are based on the frequency of occurrence of outcomes in an experiment. Furthermore, it must be possible to repeat the experiment under the same conditions. The subjective Bayesians on the other hand are willing to accept numbers based on the subjective intuition of experts of a domain. For a theoretical foundation of this approach we refer the reader to [93]. So contrary to frequentists, subjectivists can accept a probability statement about tomorrow's weather. However, both approaches tend to coincide for regular cases. Because the subjective Bayesian approach broadens the scope of application of probability theory to, among other things, computer-assisted decision making, we follow the subjective Bayesian approach to probability theory in this thesis.

For decision problems with a small number of variables, probability theory works satisfactorily. However, as the number of variables increases, straightforward probabilistic reasoning tends to lead to computational problems. The complexity of relations between events has been behind the motivation to use computers for decision support. However, the introduction of computers, and thus the possibility of handling complex systems, did not solve the computational burden introduced by probability theory until the nineteen eighties. Before, in the early nineteen sixties, very crude approximations such as the Idiot's Bayes method were proposed.

In these approaches many assumptions about independence are made in order to achieve computational efficiency. In practice, however, these independence assumptions often are violated, resulting in large errors in the probabilities obtained by these methods.

From psychology, an attempt to model human reasoning was made by using rules of the form ‘if certain conditions are true, then a conclusion is true’ [74]. Because only the conditions have to be checked for each rule to be activated and there is a limited number of rules, this is exactly the type of formalism that can be implemented efficiently on computers. It leads to programs called rule-based expert systems. However, to model the uncertainty involved in decision making, rules alone are not sufficient. To this end, among others, the certainty factor model was introduced [100] which assigns a number to every rule to represent the validity of the conclusion of the rule. A set of instructions was defined to facilitate the combination of certainty factors in rules. The certainty factor model is not well-founded from a mathematical point of view. In practice, however, rule-based expert systems that use them seem to behave satisfactorily [13].

Research proceeded in developing rule-based systems based on the assumption that in human decision making only available information is taken into account and assumptions are made about the variables that have not been observed. This led to the development of non-monotonic logics [87]. In the nineteen eighties, graphical probabilistic models were introduced [72, 79, 118] for which efficient algorithms have been developed [72, 79, 95].

Apart from approaches based on probability theory, other approaches have been developed in order to model the different types of uncertainty which cannot be handled with probability theory. For example, fuzzy logic and possibility theory have been designed to capture linguistic vagueness [122]. To differentiate between uncertainty and ignorance, probability theory was generalized in the Dempster-Shafer theory [32, 98]. Pearl [81] makes clear that there is no use in differentiating between various sources of uncertainty for making decisions. Furthermore, it has been argued that statistics is the only sound formalism to model uncertainty when one is willing to accept a very small rational set of axioms [27]. As a result, it can be shown that with any other formalism that is not a proper generalization of probability theory it is possible to arrive at irrational decisions. An overview of various formalisms and their relationship can be found in [99]. We will not elaborate on this issue in this thesis.

1.2 Bayesian Belief Networks

Bayesian belief networks are known by names such as *causal graphs* [72], *causal networks* [114], *belief networks* [79], *recursive models* [118], *probabilistic networks* [23] or permutations of two or three of these terms. We will adopt the term *Bayesian belief networks* in this thesis.

Let V be a set of variables. Then, a Bayesian belief network B over V is a pair (B_S, B_P). B_S is a directed acyclic graph with a node for each variable $v \in V$, called the *network structure*. Informally speaking, an arc between two nodes in the graph represents an influence from the node at the tail on the one at the head. B_P is a

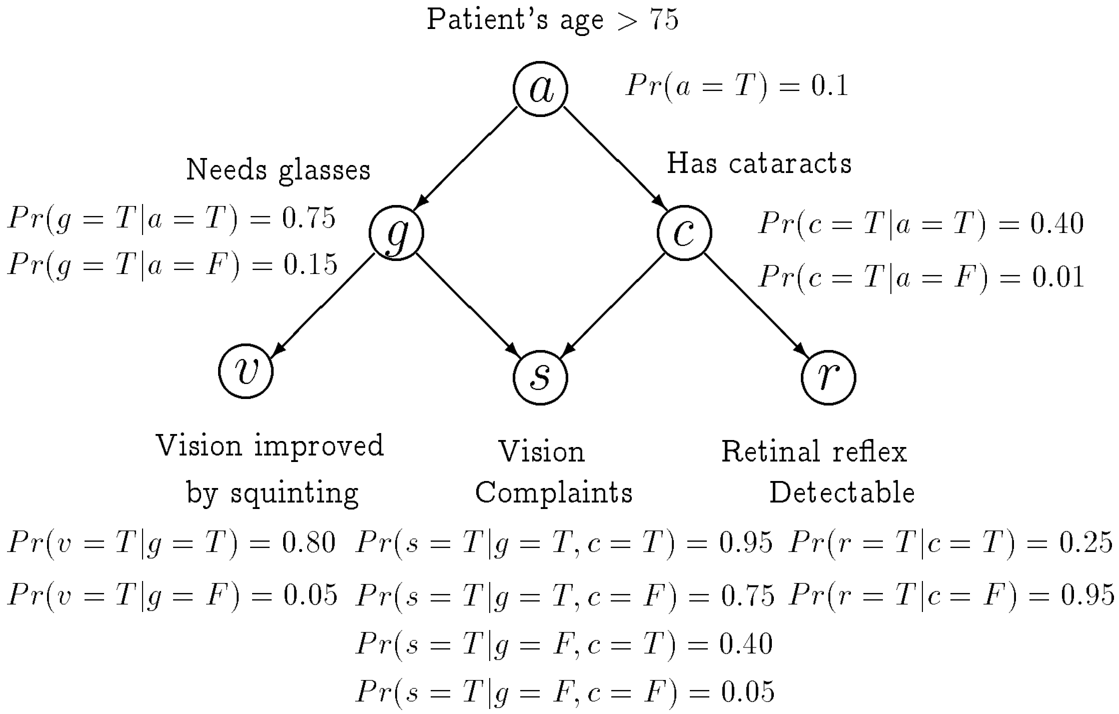


Figure 1.1: An example of a Bayesian belief network.

set of *assessment functions*, one for each variable v in V, defining a conditional probability distribution of the variable given the variables that are its parents in B_S. These functions quantify the strength of the dependencies between variables connected with an arc in the graph.

Figure 1.1 shows an example of a Bayesian belief network taken from [57]. It models the following binary variables: the patient's age is larger than 75 (a), the patient needs glasses (g), the patient has cataracts (c), the patient's vision is improved by squinting (v), the patient complains of poor vision (s), and the patient's retinal reflex is detectable (r). The presence of the arc between a and g in the network structure implies that a and g are directly dependent, relative to the other variables. The absence of an arc between a and s implies that a and s are dependent through g and c.

The strength of the dependencies between variables is quantified by the assessment functions. For example, the probability of g being true given that a is true, shown in Figure 1.1 as $Pr(g = T|a = T)$, is 0.75. Note that all conditional probabilities of a variable being false given its parents can be deduced from the conditional probabilities of the variable being true. Therefore, they have been omitted from the figure.

Together, the assessment functions of a Bayesian belief network define a unique joint probability distribution over V that agrees with the independencies represented by the network structure. Note that due to the independencies, far fewer probabilities need to be specified than with an exhaustive listing of the joint probability distribution. It is this joint probability distribution represented by a Bayesian belief network that we can use for decision support; we can enter values of observed variables and calculate the probabilities of the other variables given the evidence. For example, if we know that the patient's age does not exceed 75 and the patient complains about vision, we can calculate the probability $Pr(g = T|a = F, s = T)$ that the patient needs glasses given these observations. This process is called *inference*. The efficiency of inference algorithms is based on the presence of conditional independence.

In domains as different as medical diagnosis [4, 5, 54] and oil market prediction [2], Bayesian belief networks have been applied successfully. This indicates their practical use. In fact, the number of applications has been increasing dramatically the last few years[1].

1.3 Life Cycle of a Bayesian Belief Network

A Bayesian belief network has a qualitative part represented by the network structure, and a quantitative part, represented by the assessment functions. Apart from the definition of the domain, both these parts have to be specified to obtain a Bayesian belief network ready for usage as inference engine in a knowledge-based system. Figure 1.2 illustrates the life cycle of a Bayesian belief network; the blocks represent stages, the diamonds represent decisions, and the arrows represent what to perform next. We distinguish four stages in the life cycle of a Bayesian belief network[2]:

- definition of the domain variables,
- determination of a network structure,
- determination of the assessment functions, and
- usage in a knowledge-based system.

After every stage, there is an evaluation during which it is considered whether the previous stages were performed satisfactorily. Every time it is found that the results are not sufficient, one of the previous stages has to be passed through. Otherwise, one can proceed with the next stage. We call the repeated passing through stages and evaluations running through the *build-test cycle*. We like to stress here that the evaluations are crucial: errors made early in the life cycle can be corrected later only at large costs.

We will now examine the several components of Figure 1.2 more closely. First of all, the variables in the domain need to be fixed. In general, it is very difficult to determine the domain variables automatically, except possibly in cases where there is a technical specification at hand, such as a chip-design. Commonly, the only way to determine the domain variables is by elicitation from a domain expert. It is very important to give a precise specification of the variables and their values. If this is not performed properly, misunderstandings may occur easily; the meaning of a variable may dramatically differ between various experts, and also between them

[1] A continuously growing list of applications with Bayesian belief networks is available by anonymous ftp from `research.microsoft.com:/pub/dtg/bn-apps.ps`.

[2] In the life cycle as used in traditional software engineering other stages such as introduction of the product and maintenance are distinguished. Since these stages are out the scope of this thesis, we do not consider them explicitly.

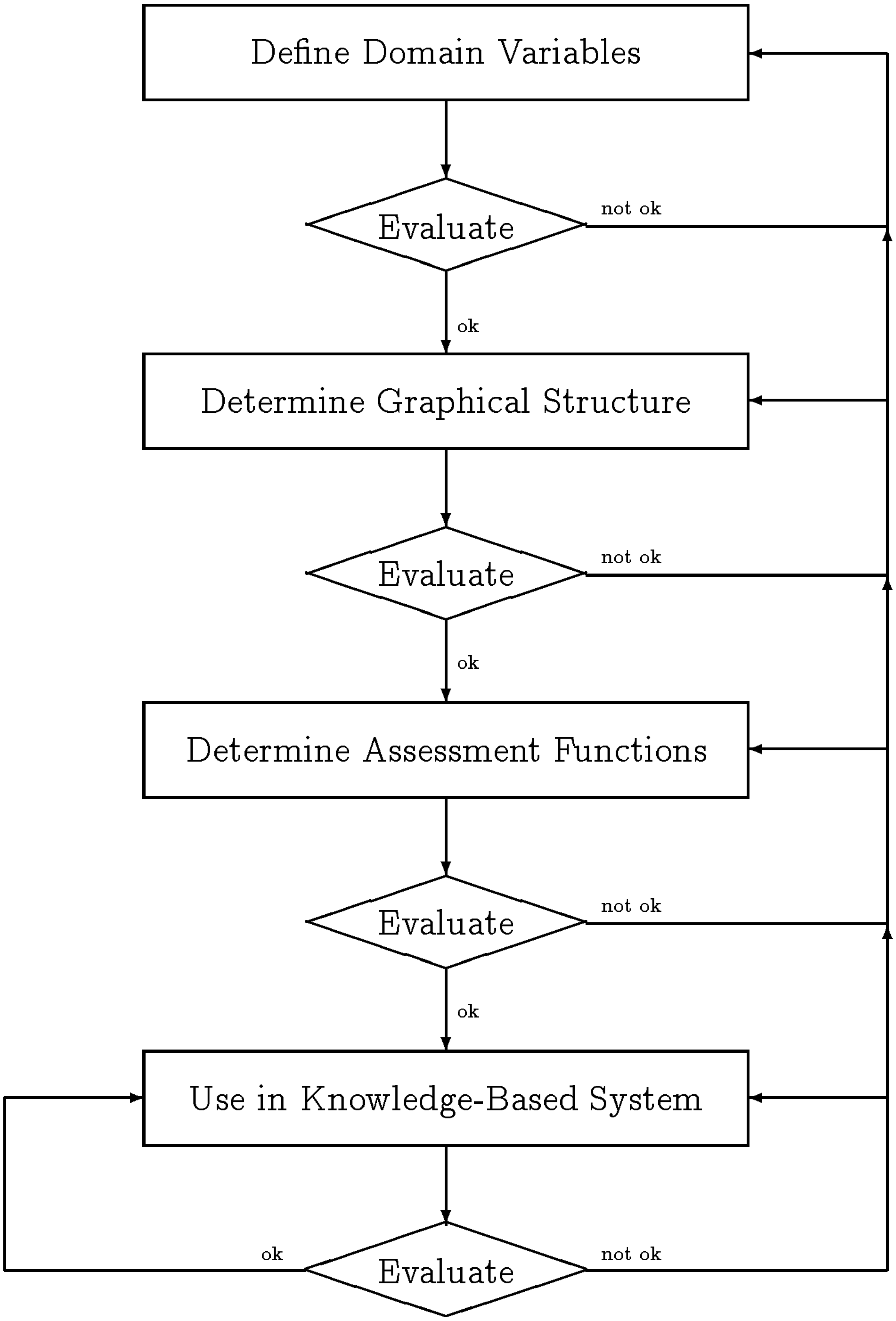

Define Domain Variables
Evaluate
not ok
ok
Determine Graphical Structure
Evaluate
not ok
ok
Determine Assessment Functions
Evaluate
not ok
ok
Use in Knowledge-Based System
ok
Evaluate
not ok

Figure 1.2: Life cycle of a Bayesian belief network.

and the users of the knowledge-based system. In this thesis, we assume that the set of variables and their domains have been chosen properly and that no further verification is necessary.

If the set of domain variables is agreed upon by all experts involved, we can proceed with determining the network structure. Causality is often used as heuristic for this task: a domain expert is asked to list all direct causes for each variable. The variables that are causes of a variable v form the parent set of v in the network structure. Care must be taken not to introduce cycles during this construction. If cycles occur, we can impose a restriction on the ordering of the variables and let the expert select for each variable v the smallest set of variables among variables that are lower ordered than v that make v conditionally independent of its predecessors. This selection mechanism guarantees a network structure without cycles. Alternatively, temporal and dynamic Bayesian networks [78, 65] may be used to avoid cycles.

After a network structure has been established, it should be carefully checked if the network structure captures all dependencies between variables. We can perform this check by confronting the domain expert with independency statements that are consequences of his assignment of parent sets. These independency statements are read from the network structure using a graphical criterion known as d-separation[3]. However, it is also possible to read statements about dependence from the network structure, as we will demonstrate in Chapter 3. If the experts do not agree on the submitted independency and dependency statements, the network structure has to be reconsidered.

Knowledge acquisition from experts is difficult because many domains are ill-understood and experts have problems in making their knowledge explicit. These domains are suitable for the application of Bayesian belief networks. Further, in some domains there are few, if any, experts.

By exploiting databases, the construction time of Bayesian belief networks may be considerably decreased. Automatically constructed Bayesian belief network may be used directly for inference (if no expert is available for evaluation of the Bayesian belief network), or serve as starting point of the build-test cycle where they are further improved by an expert. Thus, the length of the build-test cycle may be considerably shortened. The structure of an automatically constructed Bayesian belief network may give insight in the dependence of the variables in the domain, which may be applied in automated discovery of dependency relationships. Scientific hypotheses may be tested with the same method; the probabilities of the dependency structures representing causal relations of various hypotheses can be calculated.

A lot of research effort has been invested in the design of methods for learning Bayesian belief networks, from different perspectives such as computer science [24, 44], statistics [102, 120], and philosophy [47, 103]. We will review these methods in Section 1.4.

After the network structure of the Bayesian belief network has been obtained, we can define its associated assessment functions. Unfortunately, human experts are very ill-equipped for the task of probability estimation. Furthermore, the number of parameters to be determined to define an assessment function of a variable u

[3] The notion of d-separation is very important for the theory of Bayesian belief networks. Because the definition of d-separation is rather technical, we return to it in Chapter 2.

grows exponentially with the number of parents of u in the network structure. This implies that when these numbers are elicitated from an expert, it may be a very time-consuming and therefore expensive task to define the assessment functions. Alternative models to describe the relationship between a variable and its parents in the network structure have been proposed for which fewer parameters need to be specified, like the noisy-or model [81] and its generalizations [33, 56, 105].

The probabilities also may be obtained from a database of cases over the domain [24]. This is not only much faster than elicitation from experts, but it also has the advantage of returning 'objective' numbers claimed to be the only valid probabilities from a frequentist point of view. The assessment functions so defined should be checked carefully, because it may very well be possible that some configurations of variables do not appear in the database. As a result, the estimates are based on very few data-points and large errors in the estimates may occur. Another approach is to split the database in two parts; one part is used to learn the assessment functions and the other part is used for evaluation.

Once a Bayesian belief network is fully defined, it can be used in knowledge-based systems. Diagnosis, planning, and control are but a few of the tasks that can be supported with a Bayesian belief network. A lot of research is focussed on decreasing the computational complexity of algorithms for these tasks. Every time a Bayesian belief network is used in a knowledge-based system, new cases become available that can be stored in a database for validation of the network [68, 76, 102].

In this thesis, we contribute to the last three of the stages in the life cycle of a Bayesian belief network and to the evaluation of graphical structures.

1.4 Previous Work on Network Structure Learning

In network-structure learning we distinguish three main streams of work: entropy-based methods, independency-statement based methods, and Bayesian and minimum description length-based methods. We will consider these methods separately.

Entropy-Based Methods

The first algorithms developed for learning network structure for Bayesian belief networks are based on entropy. Entropy can be considered as a non-negative measure of information in a distribution. The higher the entropy, the less informative the distribution Pr. Entropy-based learning of Bayesian belief networks amounts to selecting a Bayesian belief network that represents a distribution with a low entropy: the network is selected by choosing a network structure and then estimating the assessment functions from a database. A related measure that gives the distance between two distributions Pr and Pr' over V, is the *divergence* or *cross entropy*. The larger the cross entropy, the more Pr and Pr' differ. Divergence-based learning of Bayesian belief networks amounts to selecting a network with a large divergence compared to a current network, thus visiting a sequence of networks.

Chow and Liu [20] are considered to have developed the first method for constructing network structures. They only consider trees, so all nodes have at most one parent. The main idea is to compare different distributions over two variables in the domain that are estimated from a database. In the first distribution, the two variables are considered dependent. In the second distribution, they are taken to be independent. The cross entropies of those two distributions over all pairs of nodes are calculated this way. An undirected graph is formed by starting with a graph without edges and adding an edge between the two nodes with maximal cross entropy. Next, an edge is added which has maximal cross entropy associated but does not introduce a cycle in the graph. This process is repeated until no more edges can be added. The final step consists of assigning directions to the edges in order to form a tree. Since only tree-like Bayesian belief networks can be recovered with this method, application is restricted to a small area.

The primary goal of Chow and Liu was not to construct Bayesian belief networks, since this formalism was not known at that time, but to approximate a joint probability distribution over a set of variables V by the product of (conditional) distributions over two variables. This idea has been generalized by Ku and Kullback [66], who allowed distributions over any number of variables to approximate the joint probability distribution over the domain. However, the required number of cases to get a reliable estimate of the divergence grows exponentially in the number of variables in the lower-order distributions. Further, for a set of n variables, the number of cross entropies to calculate is $\binom{n}{k}$, where k the number of variables in the lower-order distribution. This number grows exponentially in k. These practical flaws render the method impractical.

In the social sciences, *log-linear models* and their graphical interpretation gained popularity in the early nineteen eighties [30]. A log-linear model is a decomposition of the logarithm of a joint probability distribution over a set of variables V as a sum of functions of subsets of V. The graphical structure associated with such a model is an undirected graph that contains an edge between two variables u and v if there is a function over a set of variables S that includes u and v. Under some conditions also a directed acyclic graph can be associated with a log-linear model [118]. Algorithms for the construction of log-linear models generally start with a model containing functions over single variables only in which the parameters are estimated from a database. Then the model is extended by adding higher-order functions that maximize the likelihood (and therefore, the entropy) or the cross entropy with the current model. Though efficient methods exist for calculating the likelihood and cross entropy, the number of log-linear models grows exponentially with the number of variables. For an excellent overview of log-linear models and their graphical interpretation, we refer the reader to Whittaker [120].

Rebane and Pearl [86] used the first step of the algorithm of Chow and Liu to recover network structures with a poly-tree topology, that is, a network structure that does not contain cycles when the direction of the arcs is ignored. For the second step, the assignment of directions to edges, a different method is employed based on the d-separation criterion. Essentially, they first search for v-nodes and their parents; a v-node u is a node with two incoming arcs from two non-adjacent nodes. After all v-nodes have been identified, they direct the other edges such that no new v-nodes arises. This algorithm works correctly as long as the underlying distribution can be described by a Bayesian belief network with a network structure that is a poly-tree. The algorithm has been used in a medical application [40].

The Kutató algorithm developed by Herskovits and Cooper [58] is an algorithm that employs a greedy search among network structures, selecting the one with the lowest associated entropy and thus the one that represents the most expressive distribution. An ordering on the variables is assumed to be available. For each node u, Kutató starts with an empty parent set. Then, the parent sets of the variables are extended, one variable at the time. To extend the parent set of a node u, from the nodes that are lower ordered than u, the node v is selected that maximally decreases the entropy of the network structure when v is added to the parent set of u thus avoiding the introduction of cycles. The node v is added to the parent set of u if indeed the entropy decreases. This process is repeated until no nodes can be found that decrease the entropy of the network structure beyond a threshold or no candidates are left. Though a large number of extra arcs can be expected when applying this method, experiments suggest that Kutató returns network structures that contain many arcs less than the original network structure [58, 57].

Conditional Independence Based Methods

One of the basic concepts in the theory of Bayesian belief networks is *conditional independence*. We say that the sets of variables X and Y are conditionally independent given Z, written $I(X, Z, Y)$, if $P(X|ZY) = P(X|Z)$ for all possible values of X, Y and Z. For example, in the joint probability distribution represented by the network in Figure 1.1, we have $I(a, g, v)$. Conditional independence in the distribution represented by a Bayesian belief network is encoded in the graphical structure and can be read from the network structure B_S using a graphical criterion called *d-separation*.

For the class of learning algorithms based on independency statements it is assumed that a network structure perfectly represents the dependencies and independencies in the domain, that is, an independency statement is represented by the network structure if and only if it is a valid independency statement for the domain. The validity of an independency statement can be checked by performing a statistical test using a database over the domain. The basic idea of this approach is as follows. Let V be a set of variables.

1. Start with a complete undirected graph over V.
2. Remove the edge between any two nodes u and v for which a set of variables $S \subseteq V \backslash uv$ can be found such that u and v are conditionally independent given S.
3. Select edges and nodes and assign a direction to the edges to form a v-node in the network structure.
4. Assign directions to the remaining edges such that a directed acyclic graph is formed.

The basic differences in the various algorithms are in the way the sets S are found and the rules of assigning directions.

Geiger, Paz and Pearl [44] proposed an algorithm to recover poly-trees based on this approach. Using the fact that in a poly-tree, two nodes u and v are not adjacent if and only if $I(u, V \backslash uv, v)$ or $I(u, \emptyset, v)$, they recover the underlying graph. If the resulting undirected graph contains a cycle, then the domain cannot be described by a Bayesian belief network with a network structure that is a poly-tree and the algorithm terminates. Otherwise, the algorithm proceeds by assigning direction to the arcs exactly in the way that this is done in the algorithm of Pearl and Rebane.

The *SGS-algorithm*, named after their inventors Spirtes, Glymour and Scheines, can learn every possible network structure [47]. From their probabilistic theory of causality it follows that every causal process that does not involve feedback can be perfectly represented by a network structure where the direction of the arcs are interpreted as causal influences. In the SGS-algorithm, the first and second step are not specified in further detail than in the framework. In the graph resulting from the second step, an edge between u and v is replaced by an arc from u to v if a node w can be found that is adjacent to v and not to u such that a set $S \subseteq V \backslash uvw$ exists for which $I(u, S, w)$ is valid. The remaining edges are given a direction such that no new v-nodes arise.

The search for a set S such that two nodes u and v are independent given S as given in the SGS-algorithm is computationally not feasible because any subset of nodes in the domain not containing u and v need to be considered. Using the observation that if in a network structure two nodes are not adjacent, then they are independent given their parents, the inventors of the SGS-algorithm proposed a more efficient algorithm [103] called the PC-algorithm. They derived that, given two nodes u and v, to find a set S such that $I(u, S, v)$ holds, only the subsets of the nodes that are adjacent in the undirected graph at any given moment during the execution of the algorithm need to be considered for S. This is sufficient, because all nodes that are parents of a node u remain neighbors of u during the execution of the algorithm. Further, they proposed an efficiency improvement by performing the search for such sets S in separate phases. In each phase, sets S are considered of a fixed cardinality. This cardinality starts at zero and is increased by one with each phase.

For determining the direction of an edge between two nodes u and v, a node w must be found that is adjacent to v and not to u such that a set $S \subseteq V \backslash uvw$ exists and $I(u, S, w)$. Spirtes et al. make clear that if such a set can be found, then any set of variables that makes u and v conditionally independent has this property. In the PC-algorithm, for the third step the sets of the second step of the algorithm are consulted.

To direct the remaining edges in step four, they introduced two simple rules. The first rule is that if $u \rightarrow v$ is already an arc and $v \Leftrightarrow w$ is an edge in the graph, then this edge is directed towards w. The second rule is that if there is a directed path from u to v and there is an edge between u and v, then this edge is directed towards v. In the graph returned by the PC algorithm some edges may be left undirected, namely those edges for which no causal direction can be deduced. To obtain a network structure from this graph, an edge is randomly selected and randomly assigned a direction. These rules are iteratively applied until all edges have been given a direction. Unfortunately, backtracking may be necessary in order to prevent cycles in the graph.

Verma [115] gave a refinement of the fourth step in the approach by defining a set of four rules such that it is conjectured that back-tracking is not necessary anymore. Verma and Pearl [84] proposed a further optimization by preprocessing the undirected graph before starting the algorithm, thus obtaining a large reduction of the run-time of the algorithm.

An excellent overview of these algorithms and their theoretical foundation can be found in Spirtes et al. [104].

Fung and Crawford [36] applied an approach in their program 'constructor' that slightly differs from the above approach. Let B_S be the network structure that is to be recovered. Constructor starts by constructing an undirected graph in which the neighbors of a node u is the *Markov boundary* of u, that is, the set of nodes that are parents of u, children of u or parents of the children of u except u itself in B_S. Next, a heuristic search is used to select for each node a parent set from the neighbors in the obtained undirected graph.

In general, the main drawback of the algorithms based on conditional independence information is that a source needs to be available that reliably provides independency statements. Independency statements can be derived from data using statistical tests. Especially when there is a weak dependence between two variables, or when binary variables are involved, those tests require large databases to return reliable results. Together with the restriction that the independency statements represented by the network structure are exactly those in the domain, these methods are in general impractical for small databases with discrete variables.

Bayesian and Minimum Description Length Methods

With the introduction of Kutató, it became apparent that currently available techniques from machine learning could be applied to network-structure selection. The main insight was that a measure can be used for the quality of a network structure and a database of cases. For Kutató, this measure is entropy. However, new measures based on Bayesian approaches [14, 23, 101] and minimum description length approaches [8, 69, 109, 117] have been developed based on a more thorough theoretical foundation.

Measures based on a Bayesian approach start with a prior probability distribution over the space of network structures. Given a database of cases, this distribution can be updated resulting in a posterior distribution over the space of network structures. Search algorithms for selecting network structures based on this approach explore the space of network structures and return the structure with the highest probability. So, the probability of a network structure and the database can be considered a measure of the quality of the network structures.

Measures based on a minimum description length approach aim at finding an encoding in a string of the database with as few bits as possible. The basic idea is to compress the database using a probability distribution over all possible databases; the most likely databases are encoded by short messages, and the least likely are encoded by long messages, resulting in an average message length as short as possible. The distribution is represented by a Bayesian belief network. The encoding consists of two parts: the description of the Bayesian belief network, and the compressed string. The minimum description length principle selects for a given database the Bayesian belief network for which the description length is minimal.

We will elaborate on the foundations of various quality measures in Section 4.2. For an overview of the literature in this field, we refer to [15].

1.5 Previous Work on the Use of Belief Networks in Knowledge-Based Systems

Once a Bayesian belief network is completely specified and evaluated, it can be used in a knowledge-based system. The main usage of a Bayesian belief network is for calculating probabilities of a variable taking a value given the observed values of a set of variables. These observed variables are called *evidence variables*. We distinguish two types of inference algorithms: exact methods in which the probabilities are calculated without error, and approximate methods in which the probabilities are estimated based on a set of randomly generated configurations.

Exact algorithms [72, 79, 95] obtain their efficiency by exploiting the independencies represented by the network structure. However, exact inference in general with Bayesian belief networks has been proven to be NP-hard [22]. The computational complexity of exact methods strongly depends on the topology of the network. Especially when many loops occur in a network, the runtime of exact methods increases dramatically. In many applications, exact inference may not be necessary since, due to inexactness of the probability assessments in the network, approximate beliefs suffice. For an overview of exact inference methods we refer the reader to [72, 81]. We will review the various approximate methods for inference here and elaborate on them in Chapter 5.

Surprisingly, the computational complexity of approximate methods is NP-hard [29] when demanding a certain accuracy in the estimates of the probabilities. However, the runtime of approximation algorithms is linear in the number of generated configurations and variables.

The basic idea underlying all approximate algorithms is to generate a sample consisting of a set of configurations and to approximate beliefs in the various variable values by the frequency of appearance in the sample. To count the frequency of appearance, for every value of all variables a *sample score* is recorded. This score is initially zero and may be updated every time a configuration has been generated. At the end of the algorithm, the scores are normalized such that the scores associated with one variable add to unity. The normalized score of a value of a variable u is an estimate for the probability of u getting the value. The differences between the methods lie in their way of generating configurations and counting the frequency of occurrence in the sample.

Henrion [55] was the first to introduce approximate methods for inference in Bayesian belief networks with his *equiprobable sampling* algorithm. In this approximation method, the configurations are generated by randomly assigning a value to each variable that is not an evidence variable. Only the scores associated with values of variables occurring in the generated configuration are updated by increasing them with the probability that the configuration is generated according to the distribution represented by the Bayesian belief network. The problem with this method is that many configurations will be generated that hardly have any impact on the scores. On the other hand, a few of the configurations have a dominating influence on the score. Therefore, many configurations need to be generated in order to get a good approximation of probabilities.

Henrion [55] also introduced an alternative method for generating configurations, called *logic sampling.* This method returns a more representative set of configurations. First, a topological ordering on the variables is constructed. The variables are given a value following this order. The probability used to assign a value to a variable u is taken from the assessment function of u. In this way, configurations that have a large probability of occurrence will be generated more often than configurations that have a low probability of occurrence.

When there are evidence variables, all configurations in which the values generated for these variables do not coincide with the observed values will be thrown away. This is a weak part of the logic sampling algorithm since, if the probability of occurrence of the observed values for the evidence variables is very small, many configurations will be generated that do not contribute to the score.

Fung and Chang [37] observed that the fraction of configurations in which all variables but the evidence variables have the same value and therefore is not thrown away, is proportional to the probability of occurrence of the evidence variables given the values of their parents in the configuration. So, they proposed to generate values in the same way as in logic sampling but only for the variables that are not evidence variables; the evidence variables get assigned their observed values. Now the scores are increased with the probability by which the values assigned to the evidence variables would occur in the logic sampling scheme, instead of by one as in the logic sampling algorithm. Satisfactory results with this so-called *evidence weighting* or *likelihood weighing* algorithm have been reported [25, 97]. However, again a problem arises when the probability of occurrence of the observed values of the evidence variables is very small.

This motivated Shachter and Peot [97] to consider a technique widely studied and applied in statistics known as *importance sampling.* They are the first who made an explicit distinction between the distribution represented by the Bayesian belief network and the *sampling distribution*, that is, the distribution used for generating the configurations. In importance sampling, the sampling distribution is adjusted during the generation of configurations based on the so far generated configurations. First a number k of configurations is generated using likelihood weighing, so the sampling distribution equals the distribution represented by the Bayesian belief network. Now, based on the generated configurations, the assessment functions are estimated. These assessment functions are weighted with the assessment functions in the Bayesian belief network, and these new assessment functions are used to define the sampling distribution for generating the next k configurations. This process is repeated, until a sufficient number of configurations has been generated. Of course, the scores need to be adjusted accordingly. A disadvantage of importance sampling is that a lot of parameters are involved that may be chosen arbitrarily but that have a large impact of the performance. The number of configurations k after which the sampling distribution is updated may not be chosen too small because updating is computationally expensive. The weighting of the assessment functions in the Bayesian belief network and the estimated assessment functions must be taken such that both assessment functions have some influence on the sampling distribution, but none of the two must dominate completely.

All these approximate methods have difficulties with the occurrence of rare evidence. A technique that may be used to handle such evidence is *backward-sampling* [38]. In backward sampling, the variables in a parent set of a node u get assigned values all at once. This can be applied when u already has assigned a value. The probability with which the parents of u get assigned a value is proportional to the assessment function of node u in which the value of u is substituted.

The previous approximation algorithms are characterized by the complete independence of the generated configurations, except for importance sampling where there is only a slight dependence. Pearl [80] introduced a scheme in which the next configuration depends heavily on the previous configuration. An initial configuration is generated with one of the above methods. Then, in random order, the nodes are assigned a new value with a probability proportional to the product of the assessment functions in the Markov boundary of the node where the values of the configuration are substituted. Every time a variable u is assigned a value, the score of the assigned value is increased by one.

Pearl's algorithm does not have problems with rare evidence, but it does not perform well when there are very strong dependencies between variables.

Chin and Cooper [19] proposed several forms of graph modification in order to transform a Bayesian belief network in such a way that Pearl's algorithm and likelihood weighing converges faster.

In order to be able to perform an error analysis of approximation algorithms using techniques for analyzing probabilistic algorithms, Chavez [17] needed very representative but independently generated configurations. He proposed to apply Pearl's algorithm repeatedly to generate a fixed number of configurations, but to use only the last configuration for updating scores. This results in an algorithm with a very long runtime per sample but with a precise, theoretically justified estimate of the error that has not been given for the other algorithms. Recently [28], also a Bayesian error analysis for these algorithms has become available.

We refer the reader to [26] for an overview of approximation algorithms for Bayesian belief networks.

1.6 Overview of this Thesis

The organization of this thesis is as follows. In Chapter 2, we introduce terms, definitions and notations of concepts used in the rest of the thesis. In this chapter, only existing concepts and some related properties are considered; no new theory is presented.

The following three chapters can be read independently of each other. In Chapter 3, we present a theoretical framework for conditional dependence, the counterpart of conditional independence. Conditional dependence may be useful for evaluation of a network structure, that is, the second evaluation as depicted in Figure 1.2. In this chapter, we develop graphical criteria for reading statements about conditional dependence from a graph.

In Chapter 4, we investigate the problems of learning a Bayesian belief network from a database of cases. We consider both learning of network structures, in Figure 1.2 referred to as determination of the graphical structure, and learning of assessment functions, in Figure 1.2 referred to as defining the parameters of a distribution.

We investigate properties of various popular quality measures for both infinite-size and finite-size databases. Further, we consider the complexity of selecting a network structure with the highest quality. Up to now, only very simple search heuristics were proposed. We give a generalization of the known heuristics and show how to apply some general search algorithms to the task of selecting a network structure.

Learning of assessment functions can be performed by direct estimation from the database. With an alternative technique, known as smoothing, the database is explored more efficiently. We show how to incorporate smoothing into search heuristics, yielding a better estimate of assessment functions at a small computational cost. To obtain insight in the usefulness of the various techniques, we performed various experiments.

In Chapter 5, we consider approximation methods for inference in Bayesian belief networks, which is the final task in the life cycle of a network as depicted in Figure 1.2. In this chapter, we present a new method for generating configurations based on a popular statistical technique known as stratification. We show both theoretically and experimentally that our method generates the configurations faster and results in a better estimate of probabilities than other approximation methods known.

Finally, in Chapter 6 we list our main contributions, make some final concluding remarks, and point out directions for further research.

Preliminaries

In this chapter the basic concepts that are used throughout this thesis will be introduced. As the theory on probabilistic networks relies heavily on both graph theory and probability theory, two separate sections have been devoted to them. Section 2.3 addresses the concept of irrelevance, which is formalized by the notion of conditional independence. In Section 2.4, the graphical representation of conditional independence is considered. The chapter is concluded by Section 2.5, which is devoted to probabilistic networks.

2.1 Graph Theory

Some basic concepts from graph theory are reviewed in this section closely following [120]. For further information on graph theory, the reader is referred to [6, 48].

The following notational conventions will be used for sets of variables. Capital letters denote sets of variables and lower case letters denote single variables. To prevent an abundant usage of braces, sometimes u is written to denote $\{u\}$, XY to denote the set union $X \cup Y$, and uv to denote $\{u, v\}$. Set-difference, denoted by the symbol $\backslash$, is taken to have a lower priority than union and conjunction. So, $Y \cap Z \backslash Xv$ is interpreted as $(Y \cap Z) \backslash (X \cup \{v\})$. Usually, lower case letters at the end of the alphabet are used to denote single variables.

2.1 **Definition** A *graph* G is an ordered pair $G = (V(G), E(G))$, where $V(G)$ is a non-empty finite set of variables, called *nodes*, and $E(G)$ is a set of pairs of nodes (u, v), $u, v \in V(G)$, called *edges*.

There is a *directed edge* between two nodes u and v, $u, v \in V(G)$ in G, written $u \rightarrow v$ or $v \leftarrow u$, if $(u, v) \in E(G)$ and $(v, u) \notin E(G)$. If $u \rightarrow v \in E(G)$, then u is a *parent* of v and v is a *child* of u. The *parent set* π_u of u is the set of parents of u. The *child set* σ_u of u is the set of children of u.

There is an *undirected edge* between two nodes u and v, $u, v \in V(G)$ in G, written $u \Leftrightarrow v$, if both $(u, v) \in E(G)$ and $(v, u) \in E(G)$. Let $u \Leftrightarrow v \in E(G)$, then u and v are *adjacent* in G. The *neighborhood* ν_u of u is the set of all adjacent nodes of u. □

The elements of $V(G)$ will be interchangely called nodes and variables. As long as the context makes clear which graph G is meant, V and E are used to denote the sets $V(G)$ and $E(G)$, respectively. Directed edges will also be called *arcs*.

2.2 **Definition** Let $G = (V, E)$ be a graph.
G is an *undirected graph* if there are only undirected edges in G.
G is a *directed graph* if there are only directed edges in G.
G is a *mixed graph* if there are directed or undirected edges in G.
G is a *simple graph* if it does not contain edges of the form (u, u). □

In the sequel, all graphs are taken to be simple.

2.3 **Definition** Let $G = (V, E)$ be a graph. A *path* in G is a sequence of nodes $u_1, u_2, \ldots, u_m$, $m \geq 1$, such that $(u_i, u_{i+1}) \in E$ for $1 \leq i < m$. The *length* of a path is the number of nodes in a path minus one. u_m is a *descendant* of u_1 in G and u_1 is a *ascendant* of u_m.

A path $u_1, u_2, \ldots, u_m$ is *simple* if all nodes in the path are distinct.

A path $u_1, u_2, \ldots, u_m$, $m > 1$, is a *cycle* if $u_1 = u_m$. □

In the sequel, we take all paths to be simple. Usually, D_u denotes the set of all descendants of u and A_u the set of all ascendants of u.

2.4 **Definition** Let $G = (V, E)$ be a graph. G is a *directed acyclic graph* if G is a directed graph and contains no cycles. □

2.5 **Definition** Let $G = (V, E)$ be a graph.

A *chain* in G is a sequence of distinct nodes $u_1, u_2, \ldots, u_m$, $m > 1$, such that $(u_i, u_{i+1}) \in E$ or $(u_{i+1}, u_i) \in E$ for $1 \leq i < m$. The *length* of a chain is the number of nodes in a chain minus one.

A v-*node* u in a chain $u_1, u_2, \ldots, u_m$ is a node u_i, $1 < i < m$, in the chain such that $u_{i-1} \rightarrow u$ and $u \leftarrow u_{i+1}$ are arcs in G. □

2.6 **Definition** Let $G = (V, E)$ be a graph. A graph $G' = (V', E')$ is a *subgraph* of $G = (E, G)$ if $V' \subseteq V$ and $E' \subseteq E$ such that for all $(u, v) \in E'$ $u, v \in V'$. □

2.7 **Definition** Let $G = (V, E)$ be a graph and let $X \subseteq V$ be a set of nodes.

The subgraph G_X *induced* by X is the graph $(X, E(X))$ where $E(X)$ is the subset of $E(G)$ obtained by deleting all edges that have at least one node not in X.

G is *complete* if for every pair of nodes $u, v \in V(G)$ $(u, v) \in E(G)$ or $(v, u) \in E(G)$.

A *clique* X in G is a subset of nodes $X \subseteq V(G)$ such that X induces a complete subgraph G_X but no superset of X induces a complete subgraph. □

2.8 **Definition** Let $G = (V, E(G))$ be a graph. The undirected graph G_E *underlying* G is the graph $G_E = (V, E(G_E))$ where $(u, v) \in E(G_E)$ and $(v, u) \in E(G_E)$ if and only if $(u, v) \in E(G)$. □

The underlying graph G_E of G is also called the *embedded graph* of G.

2.9 **Definition** Let $G = (V, E)$ be a graph.

G is *connected* if there is a chain between any pair of nodes u and v, $u, v \in V$; otherwise, G is *disconnected*.

G is a *tree* if G is a connected undirected graph without cycles.

G is a *poly-tree* if G is a directed graph for which its underlying graph is a tree.

□

2.10 **Definition** Let $G = (V, E)$ be a directed acyclic graph. A *topological ordering* $<_V$ of G is a total ordering on V such that $u \rightarrow v \in E$ implies $u <_V v$. We say that G *obeys* an ordering $<_V$ on V if $<_V$ is a topological ordering of G. □

It is well-known that there exists a topological ordering for every directed acyclic graph.

2.2 Probability Theory

Probability theory offers a sound mathematical formalism for representing uncertainty. In this section, a short introduction to basic concepts is given. For further information, the reader is referred to [121].

2.11 **Definition** Let V be a finite nonempty set of discrete variables.

The *state space* Ω_u of a variable $u \in V$ is a finite set of *values* which the variable u may adopt, where $r_u = |\Omega_u| \geq 2$. A *value* of u is any element of Ω_u.

Let X be a subset of V. The *outcome space* Ω_X of X is the Cartesian product of all state spaces of the variables in X, $\Omega_X = \mathop{\times}\limits_{v \in X} \Omega_u$. An element of Ω_X is called a *configuration* of X.

The *event space* $\mathcal{F}$ of V is the power set of Ω_V. □

Usually, elements of Ω_u are denoted by x_u, and elements of Ω_X are denoted as $x_X = (x_u, u \in X)$. For the entire set of variables V, the subscript of the outcome space Ω_V is omitted and the outcome space is written as Ω. If X is a proper subset of V, then the outcome space Ω_X is often called a *partial outcome space* of V of X; a configuration of X is then called a *partial configuration* of V of X.

To illustrate the notation of configurations, consider $V = \{u, v, w\}$ and $X = \{u, v\}$. Then $\{V = x_V\}$, $\{u = x_u, v = x_v, w = x_w\}$, and $\{X = x_X, w = x_w\}$ are alternative notations that denote that the variables in V have some value defined by x_V, x_X, etcetera. Further, $\{X = x_X\}$ denotes the set of configurations in which the variables in X are assigned the value x_X.

2.12 **Definition** Let V, Ω, and $\mathcal{F}$ be as before. A *probability distribution* Pr over V is a function $Pr : \mathcal{F} \to [0..1]$ satisfying the following conditions:

1. $Pr(A) \geq 0$ for each $A \in \mathcal{F}$,
2. $Pr(A \cup B) = Pr(A) + Pr(A)$ for all $A, B \in \mathcal{F}$ such that $A \cap B = \emptyset$, and
3. $Pr(\Omega) = 1$.

Probability distributions for which $Pr(A) > 0$ for all $A \in \mathcal{F}$, $A \neq \emptyset$, are called *positive probability distributions*. □

Note that a probability distribution Pr over V is uniquely defined by the probabilities on the single configurations of V. In this thesis, mostly probabilities of single configurations are considered and braces that are induced by set theory are omitted. For example, $Pr(V = x_V)$ and $Pr(u = x_u, v = x_v)$ is written for $Pr(\{V = x_V\})$ and $Pr(\{u = x_u, v = x_v\})$, respectively.

If $X \subset V$, then the partial configuration $x_X = (x'_u, u \in X)$ *conforms* to $x_V = (x_u, u \in V)$ if a configuration $x_{V \backslash X}$ exists such that $\{V = x_V\} = \{X = x_X, V \backslash X = x_{V \backslash X}\}$. Likewise for $u \in V$, the value x_u *conforms* to x_V for u if a configuration $x_{V \backslash u}$ exists such that $\{V = x_V\} = \{u = x_u, V \backslash u = x_{V \backslash u}\}$.

In the sequel, formulas containing more than one partial configuration over subsets of variables of V will be taken to conform to the configuration x_V. For example, if $V = X \cup Y$, then the formula $\forall x_V \in \Omega, Pr(V = x_V) = Pr(X = x_X) \cdot Pr(Y = x_Y)$ implies that x_X conforms to x_V and x_Y conforms to x_V.

2.1 **Lemma** Let V, Ω, and $\mathcal{F}$ be as before, and let Pr be a probability distribution over V. Let X and Y be proper subsets of V such that $V = XY$ and $X \cap Y = \emptyset$. Let Ω_X and Ω_Y be the outcome spaces of X and Y respectively, and let $\mathcal{F}_Y$ be the event space of Y. Then the function $Pr' : \mathcal{F}_Y \to [0 \ldots 1]$ defined as

$$Pr'(Y = x_Y) = \sum_{x_X \in \Omega_X} Pr(X = x_X, Y = x_Y).$$

for all $x_Y \in \Omega_Y$ is a probability distribution over Y. □

The probability distribution Pr' from Lemma 2.1 is usually referred to as a *marginal probability distribution* or a *marginal* for short. To distinguish the distribution Pr from the marginal distributions, it is called the *joint probability distribution*. For ease of notation, we use Pr to denote the joint probability distribution as well as the marginal distributions derived from it. The argument to Pr will make clear which distribution is under consideration.

2.13 **Definition** Let V and $\mathcal{F}$ be as before, and let Pr be a joint probability distribution over V. Let $X \in \mathcal{F}$. Then, for each $Y \in \mathcal{F}$, with $Pr(Y) > 0$, the *conditional probability* of X given Y, written $Pr(X|Y)$, is

$$Pr(X|Y) = \frac{Pr(X \cap Y)}{Pr(Y)}.$$

□

Note that conditional probabilities $Pr(X|Y)$ are only defined in the case that $Pr(Y) > 0$. In the sequel, conditional probabilities are taken to be defined. If $Y = \emptyset$, $Pr(X|Y)$ is taken to be $Pr(X)$. It is easily seen that the function $Pr(X = x_X|Y = x_Y)$ defines a probability distribution which we will refer to as a conditional probability distribution.

For conditional probabilities, the same notations as for probability distributions are used. A very useful property of probability distributions that immediately follows from Definition 2.13 is the *chain rule* stated in the following theorem.

2.1 **Theorem** Let V and Ω be as before, with $V = \{u_1, \ldots, u_n\}$. Let Pr be a joint probability distribution over V. Then,

$$\begin{aligned} & Pr(V = x_V) = \\ & \quad Pr(u_n = x_n|u_1 = x_1, \ldots, u_{n-1} = x_{n-1}) \cdots Pr(u_2 = x_2|u_1 = x_1) \cdot Pr(u_1 = x_1). \end{aligned}$$

for all $x_V \in \Omega$. □

2.14 **Definition** Let V and Ω be as before and let Pr be a joint probability distribution over V. Let X, Y, and Z be subsets of V. Then, X is *conditionally independent* of Y given Z in Pr, if

$$Pr(X = x_X|Y = x_Y, Z = x_Z) = Pr(X = x_X|Z = x_Z),$$

for all $x_V \in \Omega$. □

Other definitions of conditional independence exist in the context of other formalisms, but these are out of the scope of this thesis.

2.3 Conditional Independence

Conditional independence is the formalization of irrelevance. It can be considered to be one of the key concepts in different calculi in artificial intelligence [107]. Independency models are useful in studying conditional independence.

2.15 **Definition** Let V be a set of variables. An *independency statement* is a statement of the form $I(X, Z, Y)$ where X, Y, and Z are disjoint subsets of V. An *independency model* M^I over V is a set of independency statements. □

A statement $I(X, Z, Y)$ should be read as X is independent of Y given Z. In the sequel, we take that $I(X, Z, \emptyset)$ is in any independency model for X and Z disjoint subsets of V. In the literature, the term dependency model is often erroneously used instead of the term independency model [81, 107]; an *in*dependency model contains information about *in*dependencies and not necessarily about dependencies. In Chapter 3, we will return to this observation.

Through conditional independence, every probability distribution Pr induces an independency model M^I.

2.16 **Definition** Let V be a set of variables, and let Pr be a joint probability distribution on V. The independency model M^I *induced* by Pr is the set of independency statements $M^I = \{I(X, Z, Y) | \forall_{x_X \in \Omega_X, x_Y \in \Omega_Y, x_Z \in \Omega_Z} Pr(X = x_X, Y = x_Y | Z = x_Z) = Pr(X = x_X | Z = x_Z) \cdot Pr(Y = x_Y | Z = x_Z), Pr(Z = x_Z) > 0\}$. □

2.17 **Definition** Let V be a set of variables. The *semi-graphoid axioms* are the following rules:

$$\begin{array}{lll}
\textit{symmetry} & I(X, Z, Y) & \Leftrightarrow I(Y, Z, X), \\
\textit{decomposition} & I(X, Z, WY) & \Rightarrow I(X, Z, W), \\
\textit{weak union} & I(X, Z, WY) & \Rightarrow I(X, ZW, Y), \text{ and} \\
\textit{contraction} & I(X, ZW, Y) \wedge I(X, Z, W) & \Rightarrow I(X, Z, WY),
\end{array}$$

for any disjoint $W, X, Y, Z \subseteq V$. The *graphoid axioms* are the semi-graphoid axioms together with the following rule:

$$\begin{array}{lll}
\textit{intersection} & I(X, ZW, Y) \wedge I(X, ZY, W) & \Rightarrow I(X, Z, WY).
\end{array}$$

for any disjoint $W, X, Y, Z \subseteq V$. □

The previous definitions give rise to a classification of independency models.

2.18 **Definition** Let V be a set of variables, and let M^I be an independency model over V.

M^I is *semi-graphoid* if M^I is closed under the semi-graphoid axioms.

M^I is *graphoid* if M^I is closed under the graphoid axioms.

M^I is *stochastic* if a joint probability distribution Pr over V exists such that Pr induces M^I.

M^I is *positive stochastic* if a positive joint probability distribution Pr over V exists such that Pr induces M^I. □

The relationship between these types of independency model is given in the following theorem.

2.2 **Theorem** Every stochastic independency model is semi-graphoid. Every positive stochastic independency model is graphoid. □

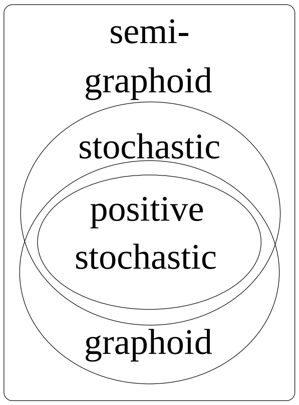


Figure 2.1: The relationship between the various classes of independency models.

A proof of the theorem can be found in [31] or in [106][1]. The relationship between the various classes of independency models is illustrated in Figure 2.1. Note that there are independency models that are both stochastic and graphoid but that are not positive stochastic. For example, let $V = \{u\}$, $\Omega_V = \{0, 1, 2\}$ and $Pr(V = 0) = Pr(V = 1) = \frac{1}{2}$, $Pr(V = 2) = 0$. Then, the independency model induced by Pr is stochastic and graphoid, but not positive stochastic.

2.4 Graphical Representations

Undirected graphs and directed acyclic graphs are a powerful means for representing independency models [30, 71, 82, 119, 120]. The representation of independency statements by an undirected graph is defined through the concepts of blocking and separation.

2.19 **Definition** Let $G = (V, E)$ be an undirected graph.

A path $u_1, u_2, \ldots, u_m$, $m > 1$ in G is *blocked* by a set of nodes X, $X \subseteq V$, if the path has at least one node in common with X. □

2.20 **Definition** Let $G = (V, E)$ be an undirected graph. Let X, Y, Z be disjoint subsets of V.

Z *separates* X from Y in G, written $\langle X, Z, Y\rangle_G$, if every path in G from a node in X to a node in Y is blocked by Z. □

[1] The theorem holds for independency statements over non-disjoint subsets as well if it is assumed that $I(X, Z, Z)$ [81].

A statement of the form $\langle X, Z, Y\rangle_G$ is called a *separation statement*. In the sequel, we omit the subscript G as long as the context makes clear which graph G is considered.

The following property of separation is known as the global Markov property.

2.3 **Theorem** Let V be a set of variables. Let $G = (V, E)$ be an undirected graph. Let $u \in V$ and ν_u be the neighborhood of u. Then, for every $u \in V$, $\langle u, \nu_u, V\backslash u\nu_u\rangle_G$. □

A proof of the theorem can be found in [81]. This theorem states that in an undirected graph, every node u is separated by its neighbors from all other nodes of the graph. An independency model can be associated with an undirected graph.

2.21 **Definition** Let V be a set of variables. Let $G = (V, E)$ be an undirected graph. The independency model M_G^I over V corresponding to G is the set of independency statements $I(X, Z, Y)$ such that $\langle X, Z, Y\rangle$ in G. □

Similar concepts of blocking and separation are defined for directed acyclic graphs.

2.22 **Definition** Let $G = (V, E)$ be a directed acyclic graph.

A chain $u_1, u_2, \ldots, u_m$, $m > 1$ in G is *blocked* by a set of nodes X, $X \subseteq V$, if there is a triple u, v, w of consecutive nodes in the chain such that one of the following conditions holds:

1. $u \rightarrow v \in E$ and $v \rightarrow w \in E$ and $v \in X$, or
2. $u \leftarrow v \in E$ and $v \rightarrow w \in E$ and $v \in X$, or
3. $u \rightarrow v \in E$ and $v \leftarrow w \in E$ and v nor its descendants D_v are in X. □

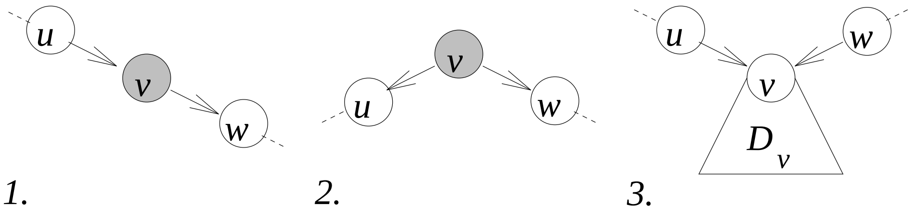


Figure 2.2: Conditions under which a chain is blocked.

The three conditions under which a chain may be blocked are illustrated in Figure 2.2.

2.23 **Definition** Let $G = (V, E)$ be a directed acyclic graph. Let X, Y, Z be disjoint subsets of V.

Z *d-separates* X from Y in G, written $\langle X, Z, Y\rangle_G$, if every chain in G between a node in X and a node in Y is blocked by Z. □

In the sequel, we will omit the subscript G from a separation statement as long as it is clear from the context which graph G is considered. We will use the term separation instead of d-separation. For separation in directed acyclic graphs, also a global Markov property is known.

2.4 **Theorem** Let V be a set of variables. Let $G = (V, E)$ be a directed acyclic graph. Then, $\langle u, \pi_u, V \backslash u \pi_u D_u \rangle_G$ for any $u \in V$. □

A proof of the theorem can be found in [81]. This theorem states that in a directed acyclic graph, every node u is separated by the parent set of u from all other nodes that are non-descendants.

2.24 **Definition** Let V be a set of variables. Let $G = (V, E)$ be a directed acyclic graph. The independency model M^I_G over V corresponding to G is the set of independency statements $I(X, Z, Y)$ such that $\langle X, Z, Y \rangle$ in G. □

2.5 **Theorem** Let V be a set of variables. Let $G = (V, E)$ be an undirected graph or a directed acyclic graph. Let M^I_G be the independency model corresponding to G. Then, M^I_G is closed under the graphoid axioms. □

Separate proofs of this theorem for undirected graphs and directed acyclic graphs can be found in [81]. The previous definitions give rise to a relation between graphs and independency models.

2.25 **Definition** Let V be a set of variables. Let M^I be an independency model over V and let $G = (V, E)$ be a graph.

G is a *dependency map*, or *D-map*, of M^I if $I(X, Z, Y) \in M^I$ implies $\langle X, Z, Y \rangle_G$.

G is an *independency map*, or *I-map*, of M^I if $\langle X, Z, Y \rangle_G$ implies $I(X, Z, Y) \in M^I$.

G is a *minimal I-map* of M^I if G is an I-map of M^I and no proper subgraph $G' = (V, E')$ of G is an I-map of M^I.

G is a *perfect map*, or *P-map*, of M^I if G is both an I-map and a D-map of M^I. □

In the sequel, often the independency model induced by a joint probability distribution Pr is considered, and a graph G will then be called a D-map, I-map, or P-map of Pr to denote that G is a D-map, I-map, or P-map, respectively, of the independency model induced by Pr.

2.26 **Definition** Let V be a set of variables and let M^I be a graphoid independency model over V.

An *independency base* L^I over M^I is a subset of M^I such that for each $u \in V$, the set L^I contains exactly one independency statement of the form

$$I(u, h_u, V \backslash u h_u)$$

and no others, where $h_u \subseteq V \backslash u$, such that $I(u, h_u, V \backslash u h_u)$ is in M^I and for any proper subset S of h_u, $I(u, S, V \backslash u S)$ is not in M^I. □

A set h_u in an element of the independency base L^I is called a *boundary* of u in L^I. A boundary h_u is constructed from the independency model M^I by initially setting h_u to $V \backslash u$ and then removing variables v from h_u for which $I(u, V \backslash uv, v)$ is in M^I.

An undirected graph $G = (V, E)$ is associated with the identified independency base by letting $u \Longleftrightarrow v \in E$ if and only if v is in the h_u. By this construction, the neighborhood ν_u in G is equal to the set h_u. Therefore, we will refer to h_u as the neighborhood of u and we will write ν_u instead of h_u. The constructed graph is a minimal I-map of M^I [81]. The independency model M^I_L associated with an independency base L^I is the closure of L^I under the independency axioms.

2.27 **Definition** Let V be a set of variables, let M^I be a graphoid independency model over V, and let $<_V$ be a total ordering on V.

A *causal input list* $L^I_{<_V}$ over M^I is a subset of M^I such that for each $u \in V$, the set $L^I_{<_V}$ contains one and only one independency statement of the form

$$I(u, p_u, V_u \backslash p_u)$$

and no others, where $V_u = \{v | v \in V, v <_V u\}$ and $p_u \subseteq V_u$ such that $I(u, p_u, V_u \backslash p_u)$ is in M^I and for any proper subset S of p_u, $I(u, S, V_u \backslash S)$ is not in M^I. □

A causal input list is constructed from an independency model M^I by starting with a set $p_u = V_u$ for each node $u \in V$. Then for each node $v \in p_u$, v is deleted from p_u if $I(u, V_u \backslash v, v)$ is in M^I.

A directed acyclic graph $G_{<_V} = (V, E)$ is associated with a causal input list $L^I_{<_V}$ as follows. G contains an arc $u \rightarrow v$ if and only if u is in the set p_u of $L^I_{<_V}$. By construction, the parent set π_u is equal to p_u. Therefore, we will refer to p_u as the parent set of v and write π_u instead of p_u. The directed acyclic graph obtained is a minimal I-map of M^I [82]. The independency model M^I_L associated with a causal input list $L^I_{<_V}$ is the closure of $L^I_{<_V}$ under the independency axioms.

Unlike for undirected graphs, an independency model need not be uniquely defined by a given directed acyclic graph; the independency model represented by the directed acyclic graph $G = (\{u, v\}, u \rightarrow v)$ is the same as the independency model represented by $G' = (\{u, v\}, u \leftarrow v)$.

2.28 **Definition** Let $G_1 = (V, E(G_1))$ and $G_2 = (V, E(G_2))$ be directed acyclic graphs.

G_1 and G_2 are *equivalent* , denoted as $G_1 \equiv G_2$, if $M^I_{G_1} = M^I_{G_2}$. □

The equivalence of directed acyclic graphs can easily be checked using the following property.

2.6 **Theorem** Let $G_1 = (V, E(G_1))$ and $G_2 = (V, E(G_2))$ be directed acyclic graphs, and let $G'_1 = (V, E(G'_1))$ and $G'_2 = (V, E(G'_2))$ be their underlying graphs, respectively.

Then, G_1 and G_2 are equivalent if and only if the following conditions hold for all $u, v, w \in V$:

1. $u \rightarrow v, v \leftarrow w \in E(G_1)$ and $u \Leftrightarrow w \notin E(G'_1)$ if and only if $u \rightarrow v, v \leftarrow w \in E(G_2)$ and $u \Leftrightarrow w \notin E(G'_2)$, and
2. $u \Leftrightarrow v \in E(G'_1)$ if and only if $u \Leftrightarrow v \in E(G'_2)$. □

A proof can be found in [82]. The two conditions for the equivalence of two directed acyclic graphs are illustrated in Figure 2.3.

Directed acyclic graphs can be transformed using the reversal of arcs.

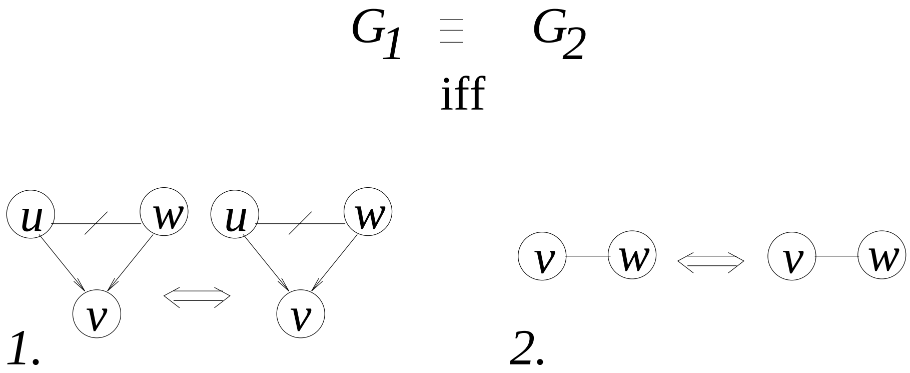


Figure 2.3: Conditions for equivalence of two directed acyclic graphs.

2.29 **Definition** Let V be a set of variables and let G be a directed acyclic graph over V. Let u, v be two nodes in V such that there is no path from v to u nor from u to v with the possible exception of the path $u \to v$.

Then, *arc-reversal* is an operation on the nodes u and v and G that results in a directed acyclic graph G' over V which equals G if $u \notin \pi_v$ and otherwise it has parent sets

$$\pi'_w = \begin{cases} \pi_w & \text{if } w \neq u \text{ and } w \neq v, \\ \pi_u \cup \pi_v \cup v & w = u, \\ \pi_v \cup \pi_u \backslash u & w = v. \end{cases}$$

□

We write $G' = arcr(G, u, v)$ to denote that G' is the directed acyclic graph obtained by applying an arc-reversal on u and v in G. If $G' = arcr(G, u, v)$, then the independency model represented by G' is a subset of the independency model represented by G and no arcs can be removed from G' without destroying this property [96].

2.5 Probabilistic Networks

Probabilistic networks offer mathematically sound formalisms for representing uncertainty in knowledge-based systems. Two kinds of probabilistic networks are considered in this thesis: Markov networks and Bayesian belief networks. The main difference between the two is that the former are based on undirected graphs and the latter are based on directed acyclic graphs.

2.30 **Definition** Let V be a set of variables. A *Markov network* B over V is an ordered pair $B = (B_S, B_P)$ such that

1. $B_S = (V, E)$ is an undirected graph, called the *skeleton* of B, containing the unique cliques $C_1, \ldots, C_m$, $m \geq 1$, and
2. $B_P = \{g_j : \Omega_{C_j} \to \mathbb{R}^+ | j = 1, \ldots, m\}$ is a set of non-negative functions, called *potentials*.

□

The potentials of a Markov network uniquely define a joint probability distribution Pr over V in which the independency statements represented by the skeleton of the network hold.

2.7 **Theorem** Let V be a set of variables, and let $B = (B_S, B_P)$ be a Markov network over V as in Definition 2.30. Then,

$$Pr(V = x_V) = \alpha \cdot \prod_{j=1}^{m} g_j(C_j = x_{C_j})$$

for all $x_V \in \Omega$, where α is a normalizing constant such that $\sum_{x_V \in \Omega} Pr(V = x_V) = 1$, defines a joint probability distribution over V such that B_S is an I-map of Pr. □

A proof of this theorem has been published in [81].

2.31 **Definition** Let V be a set of variables. A *Bayesian belief network* B over V is an ordered pair (B_S, B_P) such that

1. $B_S = (V, E)$ is a directed acyclic graph, called the *network structure* of B, and
2. $B_P = \{\gamma_u : \Omega_u \times \Omega_{\pi_u} \rightarrow [0..1] | u \in V\}$ is a set of functions, called *assessment functions*.

□

The assessment functions of a Bayesian belief network uniquely define a joint probability distribution Pr over V in which the independency statements represented by the network structure of the network hold.

2.8 **Theorem** Let V be a set of variables, and let $B = (B_S, B_P)$ be a Bayesian belief network over V as in Definition 2.31. Then,

$$Pr(V = x_V) = \prod_{u \in V} \gamma_u(u = x_u | \pi_u = x_{\pi_u})$$

for all $x_V \in \Omega$, defines a joint probability distribution over V such that B_S is an I-map of Pr. □

A proof of the theorem can be found in [62].

Conditional Dependence

The concept of irrelevance is one of the key concepts in knowledge-based systems. Without making assumptions about irrelevance, reasoning in a knowledge-based system would be intractable. Irrelevance is equivalent to conditional independence in a technical context. Conditional independence relations may be represented by independency models. Conditional independence and independency models are thoroughly studied in different calculi of artificial intelligence [107].

Since independency models are very large in general, it is not practical to store them as sets. Instead, a short list of independency statements is used to represent an independency model. Using the graphoid axioms, every independency statement in the represented independency model can be derived from such a list. There are two special kinds of lists of independency statements: the independency base, and the causal input list. These types of lists are associated with undirected graphs and directed acyclic graphs, respectively. Graphical criteria are available to read from the graph independency statements that are in the graphoid closure of the independency base or causal input list. However, there are independency models that cannot be perfectly represented by a graph. For any such model, one has to make the choice of capturing too few or too many independency statements in a graphical representation. Inference algorithms in knowledge-based systems with a Markov network or Bayesian belief network obtain their efficiency by making use of the represented independencies in their network. So, as many independency statements as possible should be represented in order to have an representation as close as possible to the true independency model. On the other hand, it is better to represent too few independency statements than too many, since erroneously assumed independencies may lead to false conclusions. Therefore, we are interested in minimal I-maps, which are graphs that represent a maximum number of independency statements but not too many.

Given a minimal I-map G of an independency model M^I, the independency statements $I(X, Z, Y)$ such that X and Y are separated by Z in G are valid independency statements in M^I. However, if X and Y are not separated by Z in G, then it is not clear whether $I(X, Z, Y)$ is an element of M^I or not. In general, if an independency statement cannot be read from the graph at hand, it does not mean that the independency statement is not valid. Yet, some independency statements may be identified that are definitely not valid. In this chapter we derive criteria for identifying such independency statements. More specifically, we develop a powerful axiomatic characterization of conditional dependence and graphical criteria for reading conditional dependencies from a minimal I-map.

It should be noted that the presented theory fits in a logical framework. Because the theory on conditional independence has been developed from a statistical point of view, we adopt the historical terminology. Therefore, this section may be confusing for readers with a background in logic; for example, we use the term 'model' for what is usually called 'theory', and we use the term 'axiom' for what is usually called 'inference rule'.

In the next section, an axiomatic characterization of conditional dependence is given together with some of its properties. In Section 3.2, we investigate the dependencies that can be derived from an independency base and we define a graphical criterion to read these dependencies from the undirected graph associated with the independency base. In Section 3.3 we address the same issues for causal input lists and directed acyclic graphs.

3.1 Conditional Dependence

We are interested in a theory about conditional dependence. In this chapter, we will consider finite sets of variables only.

3.1 **Definition** Let V be a set of variables. Let X, Y, and Z be disjoint subsets of V such that $X \neq \emptyset$, $Y \neq \emptyset$. We call X and Y *conditionally dependent* given Z, written $D(X, Z, Y)$, if X and Y are not conditionally independent given Z. The statement $D(X, Z, Y)$ is called a *dependency statement*. □

The semantics of conditional dependence is determined by the formalism in which conditional independence is defined. In general, $D(X, Z, Y)$ can be interpreted as saying that knowledge of some variables in Y is relevant for knowledge of some variables in X when all variables in Z are known. From the definition, we have $D(X, Z, Y) \Leftrightarrow \neg I(X, Z, Y)$. To study the properties of sets of dependency statements, we use dependency models.

3.2 **Definition** Let V be a set of variables. A *dependency model* M^D over V is a set of dependency statements. The *complement* of M^D is the independency model $M^I = \{I(X, Z, Y) | X, Z, Y \text{ disjoint subsets of } V, D(X, Z, Y) \notin M^D\}$. □

If M^I is the complement of M^D, then we say that M^D is the complement of M^I. Just as conditional dependence is the counterpart of conditional independence, a dependency model is the counterpart of an independency model. We now classify dependency models as we have done for independency models in Chapter 2.

3.3 **Definition** Let V be a set of variables. Let M^I be an independency model over V and let M^D be the complement of M^I.
M^D is *semi-graphoid* if M^I is semi-graphoid.
M^D is *graphoid* if M^I is a graphoid.
M^D is *stochastic* if M^I is stochastic.
M^D is *positive stochastic* if M^I is positive stochastic. □

The relationship between these types of dependency model is given in the following theorem.

3.1 **Theorem** Every stochastic dependency model is semi-graphoid. Every positive stochastic dependency model is graphoid. □

$I(a,\emptyset,b)'$
$I(a,\emptyset,bd)'$
$I(a,\emptyset,d)'$
$I(a,b,d)'$
$I(a,bc,d)$
$I(a,bc,de)$
$I(a,bc,def)$
$I(a,bc,df)$
$I(a,bc,e)$
$I(a,bc,ef)$
$I(a,bc,f)$
$I(a,bcd,e)$
$I(a,bcd,ef)$
$I(a,bcd,f)$
$I(a,bcde,f)$
$I(a,bcdf,e)$
$I(a,bce,d)$
$I(a,bce,df)$
$I(a,bce,f)$
$I(a,bcef,d)$
$I(a,bcf,d)$
$I(a,bcf,de)$
$I(a,bcf,e)$
$I(a,be,d)^*$
$I(a,bef,d)^*$
$I(a,cd,e)$
$I(a,cd,ef)$
$I(a,cd,f)$
$I(a,cde,f)$
$I(a,cdf,e)$
$I(a,ce,f)$
$I(a,d,b)'$
$I(ab,cd,e)$
$I(ab,cd,ef)$
$I(ab,cd,f)$
$I(ab,cde,f)$
$I(ab,cdf,e)$
$I(ab,ce,f)$
$I(abd,ce,f)$
$I(ac,b,d)'$
$I(ac,be,d)^*$
$I(ac,bef,d)^*$
$I(acf,be,d)^*$
$I(ad,bce,f)$
$I(ad,ce,f)$
$I(af,bce,d)$
$I(af,be,d)^*$

$I(b,\emptyset,a)'$
$I(b,acd,e)$
$I(b,acd,ef)$
$I(b,acd,f)$
$I(b,acde,f)$
$I(b,acdf,e)$
$I(b,ace,f)$
$I(b,cd,e)$
$I(b,cd,ef)$
$I(b,cd,f)$
$I(b,cde,f)$
$I(b,cdf,e)$
$I(b,ce,f)$
$I(b,d,a)'$
$I(bd,\emptyset,a)'$
$I(bd,ace,f)$
$I(bd,ce,f)$

$I(c,ab,d)'$
$I(c,abe,d)^*$
$I(c,abef,d)^*$
$I(c,b,d)'$
$I(c,be,d)^*$
$I(c,bef,d)^*$
$I(cf,abe,d)^*$
$I(cf,be,d)^*$

$I(d,\emptyset,a)'$
$I(d,ab,c)'$
$I(d,abce,f)$
$I(d,abe,c)^*$
$I(d,abe,cf)^*$
$I(d,abe,f)^*$
$I(d,abef,c)^*$
$I(d,ace,f)$
$I(d,b,a)'$
$I(d,b,ac)'$
$I(d,b,c)'$
$I(d,bc,a)$
$I(d,bce,a)$
$I(d,bce,af)$
$I(d,bce,f)$
$I(d,bcef,a)$
$I(d,bcf,a)$
$I(d,be,a)^*$
$I(d,be,ac)^*$
$I(d,be,acf)^*$
$I(d,be,af)^*$
$I(d,be,c)^*$
$I(d,be,cf)^*$
$I(d,be,f)^*$
$I(d,bef,a)^*$
$I(d,bef,ac)^*$
$I(d,bef,c)^*$
$I(d,ce,f)$
$I(de,bc,a)$
$I(de,bcf,a)$
$I(def,bc,a)$
$I(df,bc,a)$
$I(df,bce,a)$

$I(e,acd,b)$
$I(e,acdf,b)$
$I(e,bc,a)$
$I(e,bcd,a)$
$I(e,bcdf,a)$
$I(e,bcf,a)$
$I(e,cd,a)$
$I(e,cd,ab)$
$I(e,cd,b)$
$I(e,cdf,a)$
$I(e,cdf,ab)$
$I(e,cdf,b)$
$I(ef,acd,b)$
$I(ef,bc,a)$
$I(ef,bcd,a)$
$I(ef,cd,a)$
$I(ef,cd,ab)$
$I(ef,cd,b)$

$I(f,abce,d)$
$I(f,abe,d)^*$
$I(f,acd,b)$
$I(f,acde,b)$
$I(f,ace,b)$
$I(f,ace,bd)$
$I(f,ace,d)$
$I(f,bc,a)$
$I(f,bcd,a)$
$I(f,bcde,a)$
$I(f,bce,a)$
$I(f,bce,ad)$
$I(f,bce,d)$
$I(f,be,d)^*$
$I(f,cd,a)$
$I(f,cd,ab)$
$I(f,cd,b)$
$I(f,cde,a)$
$I(f,cde,ab)$
$I(f,cde,b)$
$I(f,ce,a)$
$I(f,ce,ab)$
$I(f,ce,abd)$
$I(f,ce,ad)$
$I(f,ce,b)$
$I(f,ce,bd)$
$I(f,ce,d)$

Figure 3.1: The graphoid independency model M_e^I obtained by closing $\{I(a,\emptyset,\emptyset), I(b,\emptyset,a), I(c,ab,\emptyset), I(d,b,ac), I(e,cd,ab), I(f,ce,abd), I(c,be,d)\}$ under the graphoid axioms.

Proof: The properties stated follow directly from Definition 3.3 and Theorem 2.2. □

Note that the relations between the various classes of dependency model are exactly the same as the relations between the various classes of independency model as illustrated in Figure 2.1.

Figure 3.1 shows an example of an independency model. It lists the elements of a graphoid independency model over six variables $abcdef$. The independency model is constructed by starting with a short list of independency statements $\{I(a,\emptyset,\emptyset), I(b,\emptyset,a), I(c,ab,\emptyset), I(d,b,ac), I(e,cd,ab), I(f,ce,abd), I(c,be,d)\}$ and taking the closure of this list under the graphoid axioms. All trivial independency statements are omitted, that is, no independency statements of the form $I(X,Z,\emptyset)$ is listed. The figure demonstrates that even for a small number of variables, independency models may be very large sets. By construction, the independency model can be efficiently represented by only a small set of independency statements. In general, this need not be the case. In practice, however, most independency models do have this property [104]. The listed independency model will be used in subsequent examples and will be referred to as M^I_e. Like independency models, dependency models may become very large. For example, the dependency model that is the complement of the independency model M^I_e in Figure 3.1 contains 2552 elements. Instead of storing all dependency statements, a list of selected dependency statements is a more structured representation, which makes it easier to derive theoretical results. Using a set of rules, every dependency statement in the dependency model can be derived from the list. For this purpose, the following rules are defined.

3.4 **Definition** Let V be a set of variables. The *semi-dependency axioms* are the following rules:

$$\begin{array}{lll}
\textit{symmetry} & D(X,Z,Y) & \Leftrightarrow D(Y,Z,X), \\
\textit{composition} & D(X,Z,Y) & \Rightarrow D(X,Z,WY), \\
\textit{weak reunion} & D(X,ZW,Y) & \Rightarrow D(X,Z,WY), \\
\textit{extraction} & D(X,Z,WY) \wedge I(X,Z,W) & \Rightarrow D(X,ZW,Y), \text{ and} \\
\textit{extraction+} & D(X,Z,WY) \wedge I(X,ZY,W) & \Rightarrow D(X,Z,Y)
\end{array}$$

for any disjoint $W,X,Y,Z \subseteq V$. The *dependency axioms* are the semi-dependency axioms together with the following rule:

$$\begin{array}{lll}
\textit{intersection} & D(X,Z,WY) \wedge I(X,ZY,W) & \Rightarrow D(X,ZW,Y).
\end{array}$$

for any disjoint $W,X,Y,Z \subseteq V$. □

Note that for every graphoid axiom for conditional independence a complementary dependency axiom is defined. In fact, these axioms are constructed by taking the 'inverse' of the graphoid axioms; assuming that a statement on the right-hand side of a graphoid axiom is a dependency statement, we conclude that at least one of the statements on the left-hand side must be a dependency statement. As a result, for the contraction axiom two complementary axioms are defined, namely extraction and extraction+. The same would be expected for intersection. However, the two resulting rules turn out to be equivalent.

The dependency axioms can be regarded as inference rules for certain dependency models. The intuition behind symmetry is straightforward. Composition tells that if Y is dependent of X given Z, then Y together with W also is dependent of X given Z. Weak reunion is interpreted in a similar fashion; if Y is dependent of X given ZW, then also Y together with W is dependent of X when only Z is known.

The extraction axiom says that if W and Y are dependent of X given Z, and furthermore W is independent of X given Z, then X must be dependent of Y when both Z and W are known. To get an intuition of the extraction+ a same line of reasoning can be followed; if W and Y are dependent of X when Z is known, and furthermore W is independent of X when both Z and Y are known, then X must be dependent on some variables in Y and not in W when Z is known. Note that extraction+ is the only dependency axiom in which the dependency statement on the right-hand side contains less variables than the dependency statement on the left-hand side. So, a stronger statement on dependencies is obtained.

The intuition of the intersection axioms is that if W and Y are dependent of X given Z, and furthermore W is independent of X given both Z and Y, then X must be dependent of some variables in Y given both W and Z. The intersection axiom is the strongest of the dependency axioms in the sense that it gives very specific information on which variables are conditionally dependent. With a relatively weak statement about conditional dependence and conditional independence, a rather strong statement about conditional dependence can be derived.

Note that it is necessary to have knowledge of some conditional independency statements in order to apply the extraction axioms and the intersection axiom.

3.2 **Theorem** Every semi-graphoid dependency model is closed under the semi dependency axioms. Every graphoid dependency model is closed under the dependency axioms. □

Proof: We will only show that the extraction axiom $D(X,Z,WY) \wedge I(X,Z,W) \Rightarrow D(X,ZW,Y)$ holds in every semi-graphoid dependency model. The proofs for the other axioms are analogous.

Let M^D be a semi-graphoid dependency model and let M^I be the complement of M^D. Assume that $D(X,Z,WY) \in M^D$ and $I(X,Z,W) \in M^I$. By definition we have $D(X,Z,WY) \in M^D \Leftrightarrow I(X,Z,WY) \notin M^I$. Now, suppose that $I(Y,ZW,X) \in M^I$. By contraction for graphoid independency models, it follows that $I(X,Z,WY) \in M^I$. From the contradiction, we conclude that $I(X,Z,WY) \notin M^I$ from which it follows that $D(X,ZW,Y) \in M^D$. □

The semi-dependency axioms form a relatively weak set of inference rules. From now on, we shall only consider graphoid dependency models. Note that, for example, stochastic dependency models induced by positive probability distributions are graphoid. So, we still consider a large class of dependency models.

The following properties show the power of the graphoid axioms and dependency axioms for deriving new properties.

3.1 **Lemma** Let V be a set of variables, and let M^I be a graphoid independency model over V. Let X, Y, and Z be disjoint subsets of V. If $I(u, XYZ\backslash uv, v) \in M^I$ for all $u \in X$ and $v \in Y$, if and only if $I(X, Z, Y) \in M^I$. □

Proof: First we show the only if part. We begin by showing that if, for a given $u \in X$, we have that $I(u, XYZ\backslash uv, v) \in M^I$ for all $v \in Y$, then $I(u, ZX\backslash u, Y) \in M^I$. We prove this property by induction on the cardinality of Y. For $|Y| = 0$ and $|Y| = 1$, the property holds by definition. Now, assume that for some $k > 1$, the property holds for all sets Y with $|Y| = i$, $1 \leq i < k$. Consider a set $Y = \{v_1, \ldots, v_k\}$ such that $I(u, XYZ\backslash uv, v) \in M^I$ for all $v \in \{v_1, \ldots, v_{k-1}\}$. Then from the induction hypothesis we have $I(u, XZv_k\backslash u, Y\backslash v_k) \in M^I$. By applying the intersection axiom, we find $I(u, XZ\backslash u, Y) \in M^I$. This completes the induction.

By symmetry, we have $I(Y, XZ\backslash u, u) \in M^I$ for all $u \in X$, since we can apply the property above for every u. By a similar argument as above we find $I(Y, Z, X) \in M^I$ and hence $I(X, Z, Y) \in M^I$.

The if part of the lemma easily follows from weak union and symmetry. □

3.1 **Corollary** Let V be set of variables. Let M^D be a graphoid dependency model over V, and let X, Y, and Z be disjoint subsets of V. Then, $D(X, Z, Y) \in M^D$ if and only if variables $u \in X, v \in Y$ exist such that $D(u, XYZ\backslash uv, v) \in M^D$. □

Proof: The corollary states the logical negation of Lemma 3.1. □

Note that from the corollary it follows that to show that Y is dependent on X given Z, it suffices to find two variables $u \in X$ and $v \in Y$ such that u and v are dependent given all other variables $XYZ\backslash uv$.

3.2 Undirected Graphs

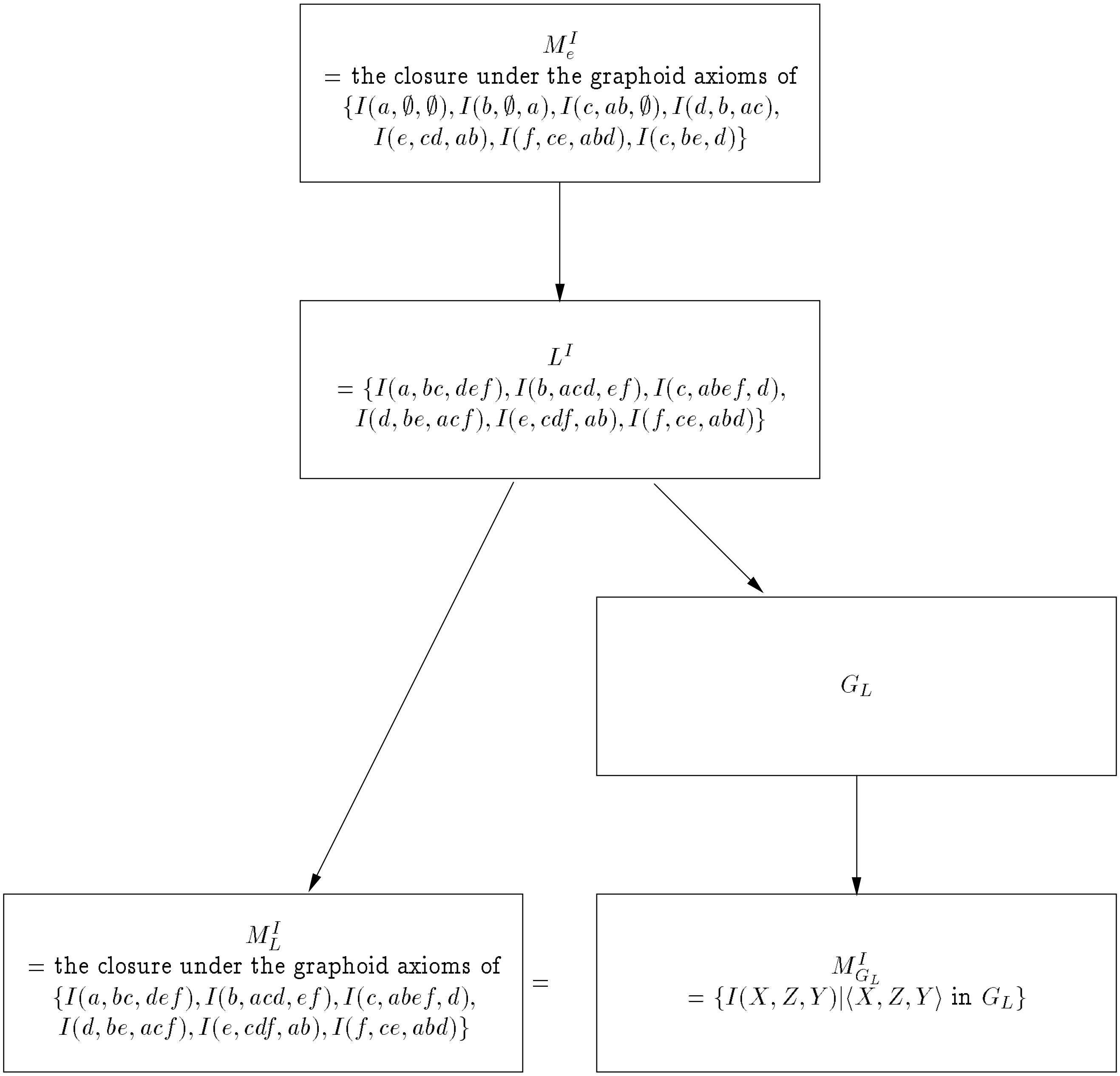
Graphs constitute a powerful representation formalism for dependency and independency models. In this section we consider undirected graphs and the way they represent independency models and dependency models. In the next section we consider directed acyclic graphs.

3.2.1 Conditional Independence in Undirected Graphs

In this subsection, we will review the relationship between a graphoid independency model, its independency base and its associated undirected graph. We will use the independency model M_e^I listed in Figure 3.1 to illustrate the relationships.

Let M^I be the independency model over a set of variables V. The independency base L^I over M^I is the set of independency statements $L^I = \{I(u, \nu_u, V\backslash u\nu_u) | u \in V\}$ as defined in Definition 2.26. So, the sets ν_u in L^I are the smallest subsets of variables of $V\backslash u$ that give full information to the variable u: if values for the variables in ν_u are known, no knowledge of values of other variables will change the probabilities for u's values.

Given the independency model M^I, the sets ν_u as described above can be constructed based on Lemma 3.1 as follows. Initially ν_u is set to $V\backslash u$ for a variable



M_e^I
$= \text{the closure under the graphoid axioms of}$
$\{I(a,\emptyset,\emptyset), I(b,\emptyset,a), I(c,ab,\emptyset), I(d,b,ac),$
$I(e,cd,ab), I(f,ce,abd), I(c,be,d)\}$

L^I
$= \{I(a,bc,def), I(b,acd,ef), I(c,abef,d),$
$I(d,be,acf), I(e,cdf,ab), I(f,ce,abd)\}$

G_L

M_L^I
$= \text{the closure under the graphoid axioms of}$
$\{I(a,bc,def), I(b,acd,ef), I(c,abef,d),$
$I(d,be,acf), I(e,cdf,ab), I(f,ce,abd)\}$

$=$

$M_{G_L}^I$
$= \{I(X,Z,Y) | \langle X,Z,Y\rangle \text{ in } G_L\}$

Figure 3.2: Relation between independency models, independency base, undirected graph, and represented independency model for M_e^I.

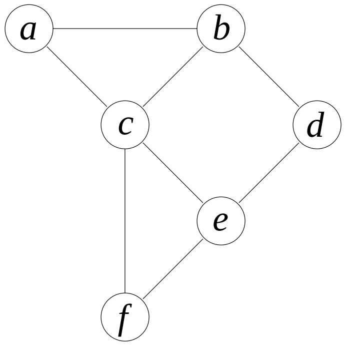


Figure 3.3: The undirected graph G_L induced by the independency base L in Figure 3.2.

$u \in V$. Then, M^I is consulted for every independency statement $I(u, V\backslash uv, v)$ with $u \in V$, $v \in V\backslash u$. If the independency statement is in M^I, then v is removed from the neighborhood ν_u. This process is repeated for every variable $u \in V$. The procedure is depicted below in pseudo code. In practical applications, a domain expert acts as an oracle that can provide answers about independency statements.

```
**Independency Base Construction for $M^I$**

for all $u \in V$ do  $\nu_u \leftarrow V\backslash u$
for all $u \in V$ do
    for all $v \in V$ do
        if $I(u, V\backslash uv, v) \in M^I$ then $\nu_u \leftarrow \nu_u\backslash v$
return $\{I(u, \nu_u, V\backslash\nu_u) | u \in V\}$
```

For example, consider the independency model M^I_e listed in Figure 3.1. Then the set ν_a is found by starting with $V\backslash a = bcdef$. As $I(a, cdef, b)$ and $I(a, bdef, c)$ are not in M^I_e, b and c cannot be removed from ν_a. But $I(a, bcef, d)$ is in M^I_e so d can be removed from ν_a resulting in $bcef$. Furthermore, as $I(a, bcdf, e)$ and $I(a, bcde, f)$ are in M^I_e, e and f can be deleted from ν_a resulting in $\nu_a = bc$. For variables a, the statement $I(a, bc, def)$ is added to the independency base. Applying this construction for the other variables gives the independency base $L^I = \{I(a, bc, def), I(b, acd, ef), I(c, abef, d), I(d, be, acf)I(e, cdf, ab), I(f, ce, abd)\}$. L^I is also listed in Figure 3.2.

Note that there is a kind of symmetry in the neighborhoods; if $u \in \nu_v$ then $v \in \nu_u$. For example, in the independency base in Figure 3.2 we have $b \in \nu_a$ and $a \in \nu_b$. The rationale behind this property is that dependence between two variables is symmetric. If a is dependent on b, then b is dependent on a.

The undirected graph G_L induced by an independency base L^I has an edge $u \Leftrightarrow v$ between u and v if and only if u is in the neighborhood ν_v of v. The undirected graph associated with the independency base listed in Figure 3.2 is shown in Figure 3.3. Since $\nu_a = bc$, there are edges $a \Leftrightarrow b$ and $a \Leftrightarrow c$ in G_L.

With the undirected graph G_L, an independency model M^I_G over V is associated containing the independency statements $I(X, Z, Y)$ if and only if X and Y are separated by Z in G_L. For example, for the graph G_L in Figure 3.3 we have that $\langle a, cd, f\rangle$ since all paths in G from a to f contain c or d. Therefore, M_G^I contains $I(a, cd, f)$. So, a graph represents an independency model through separation. As separation obeys the graphoid axioms, the represented model M_G^I is a graphoid independency model. The relationship between the graph G_L associated with the independency base L^I and the independency model M^I is given by the following theorem.

3.3 **Theorem** Let V be set of variables. Let M^I be a graphoid independency model over V and let L^I be the independency base of M^I. Let G_L be the undirected graph associated with L^I. Then, G_L is a minimal I-map of M^I. □

A proof of the theorem can be found in [83] and [81].As a consequence, we have the following property.

3.2 **Corollary** Let V be set of variables. Let M^I be a graphoid independency model over V and let L^I be the independency base of M^I. Let G_L be the undirected graph associated with L^I. Then, $M_G^I \subseteq M^I$. □

The reverse $M^I \subseteq M_G^I$ does not necessarily hold. For example, consider once more the independency model M_e^I listed in Figure 3.2 and the undirected graph G_L shown in Figure 3.3 associated with the independency base L^I of M_e^I. From M_e^I, we have that $I(a, \emptyset, b) \in M_e^I$. In G_L, however, $\langle a, \emptyset, b\rangle$ does not hold. So, $I(a, \emptyset, b) \notin M_G^I$. The independency statements from M_e^I that are not represented by G_L are marked with a prime in Figure 3.1. Note that all but sixteen independency statements are represented by G_L, indicating that undirected graphs indeed offer a powerful way of representing independency models.

The independency base L^I induces an independency model. This independency model M_L^I is the closure of L^I under the graphoid axioms. For example, let L^I be the independency base in Figure 3.2. The independency statement $I(a, bc, def)$ is in L^I so $I(a, bc, def)$ is in M_L^I. Using weak union and symmetry, we get $I(ef, bcd, a) \in M_L^I$. The independency statements $I(b, acd, ef)$ is in L^I. Using symmetry and intersection on the last two independency statements we get $I(ef, cd, ab) \in M_L^I$ and using symmetry and decomposition we have $I(a, cd, f) \in M_L^I$. So, $I(a, cd, f)$ is in M_L^I. Note that $I(a, cd, f)$ is also in the independency model M_G^I associated with the graph G_L induced by L^I. This is no coincidence, considering the following property.

3.4 **Theorem** Let V be a set of variables. Let M^I be a graphoid independency model over V. Let L^I be the independency base of M^I and let M_L^I be the graphoid closure of L^I. Let G_L be the undirected graph associated with L^I and let M_G^I be the independency model of G_L. Then,

$$M_G^I = M_L^I.$$

□

Proof: We show that $M_L^I \subseteq M_G^I$ and $M_G^I \subseteq M_L^I$.

First, we show that $M_L^I \subseteq M_G^I$ by demonstrating for all independency statements $I(X, Z, Y)$ that if $I(X, Z, Y) \in M_L^I$ then $I(X, Z, Y) \in M_G^I$. Consider any independency statement $I(u, \nu_u, V \backslash u\nu_u) \in L^I$. By construction of G_L and the definition of separation, we have that $\langle u, \nu_u, V\backslash u\nu_u \rangle_{G_L}$. So, $I(u, \nu_u, V\backslash u\nu_u) \in M^I_G$. We conclude that $L^I \subseteq M^I_G$. Since M^I_G is closed under the graphoid axioms by Theorem 2.5, every independency statement $I(X, Z, Y) \in M^I_L$ is in M^I_G. Hence $I(X, Z, Y) \in M^I_G$.

Next, we show that $M^I_G \subseteq M^I_L$ by proving for all independency statements $I(X, Z, Y)$ that if $I(X, Z, Y) \in M^I_G$ then $I(X, Z, Y) \in M^I_L$. Assume that $\langle X, Z, Y \rangle$ holds in G_L. Then for all $u \in X$ and $v \in Y$, u and v are non-adjacent. We conclude that, for all $u \in X$ the set $Y\backslash\nu_u$ is empty. So, for all $u \in X$ the statement $I(u, \nu_u, V\backslash u\nu_u) \in L^I$ can be written as $I(u, \nu_u, Y(V\backslash u\nu_u Y))$. Using decomposition we can derive $I(u, \nu_u, Y(Z\backslash\nu_u))$ and using weak union we can derive $I(u, ZX\backslash u, Y)$ for all $u \in X$. By applying Lemma 3.1, $I(X, Z, Y)$ can be derived using these statements. So, $I(X, Z, Y) \in M^I_L$. □

So, all independency statements that can be derived from an independency base L^I using the graphoid axioms are also represented in the graph induced by L^I and vice versa.

The relationships among the above concepts is depicted in Figure 3.2. In the figure, a directed arrow between two rectangles means that the object at the tail of the arrow is sufficient to construct the object at the head of the arrow.

3.2.2 Conditional Dependence in Undirected Graphs

In the previous subsection, it was shown that the representation of an independency model by an undirected graph is equivalent to the representation of this independency model by its independency base. In this subsection, we will derive a similar property for dependency models.

3.5 **Definition** Let V be a set of variables. Let M^I be a graphoid independency model over V, and let L^I be the independency base of M^I. The *dependency base* L^D of M^I is a set of dependency statements such that for each $u \in V$, L^D contains dependency statements of the form

$$D(u, \nu_u\backslash v, v)$$

for each $v \in \nu_u$, where ν_u is such that $I(u, \nu_u, V\backslash\nu_u v) \in L^I$. □

Consider once more the independency model M^I_e listed in Figure 3.1. For variable a, L^I contains the independency statement $I(a, bc, def)$. So, L^D contains the dependency statements $D(a, c, b)$, and $D(a, b, c)$. Figure 3.4 shows the entire dependency base for the independency model M^I_e listed

For the dependency base as defined above, we can show that the dependency statements that are in the dependency base indeed are in the complement of M^I, and hence are valid statements about dependence.

3.2 **Lemma** Let V be a set of variables. Let M^I be a graphoid independency model over V, and let M^D be its complementary graphoid dependency model. Let L^D be the dependency base of M^I. Then

$$L^D \subseteq M^D.$$

□

Proof: The property will be proved by contradiction. Suppose that for some $u, v \in V$, $D(u, \nu_u \backslash v, v) \in L^D$ and $D(u, \nu_u \backslash v, v) \notin M^D$. From $D(u, \nu_u \backslash v, v) \notin M^D$, we have by definition that $I(u, \nu_u \backslash v, v) \in M^I$. Now, let L^I be the independency base of M^I. By construction of L^D we know that $I(u, \nu_u, V \backslash u\nu_u)_{M^I} \in L^I$ and hence that $I(u, \nu_u, V \backslash u\nu_u)_{M^I} \in M^I$. Applying the contraction axiom to $I(u, \nu_u \backslash v, v)_{M^I}$ and $I(u, \nu_u, V \backslash u\nu_u)_{M^I}$ gives $I(u, \nu_u \backslash v, V \backslash (u\nu_u \backslash v)) \in M^I$. Now observe that if this last statement is in the independency model M^I, then ν_u is not the smallest set such that $I(u, \nu_u, V_u \backslash u\nu_u) \in M^I$. But then, L^I cannot be the independency base of M^I. From this contradiction we conclude that $L^D \subseteq M^D$. □

We can associate a dependency model with a dependency base in a similar way as we associated an independency model M^I_L with an independency base L^I.

3.6 **Definition** Let V be a set of variables. Let M^I be a graphoid independency model over V and let L^I be the independency base of M^I. Let M^I_L be the independency model associated with L^I. Let L^D be the dependency base of M^I. The dependency model M^D_L associated with L^D is the closure of L^D under the six dependency axioms, where the independency statements used are elements of M^I_L. □

Note that in applying the dependency axioms for calculating the closure of L^D, the independency statements used can be easily read from the undirected graph G_L associated with L^I, as $M^I_G = M^I_L$.

3.5 **Theorem** Let V be a set of variables. Let M^I be a graphoid independency model over V, and let M^D be its complementary graphoid dependency model. Let L^D be the dependency base of M^I and let M^D_L be its associated dependency model. Then

$$M^D_L \subseteq M^D.$$

□

Proof: By Lemma 3.2, we have that $L^D \subseteq M^D$. From Theorem 3.2 we have that any graphoid dependency model is closed under the dependency axioms. The property stated in the theorem now follows from the definition of M^D_L. □

The dependency model M^D_L can be viewed as the counterpart of the independency model M^I_L. We now take the *dependency pool* M over V as the set of all triples (X, Z, Y) where X, Y, and Z are disjoint subsets of V. For a given independency base L^I, we can divide the pool M into four disjoint sets: M^I_L, M^D_L, $M^I \backslash M^I_L$ and $M^D \backslash M^D_L$. Figure 3.5 illustrates this basic idea. Given an independency base, of the statements in the last two sets we cannot tell whether they are independency statements or dependency statements unless we have extra information.

For example, consider a memory chip with eight bits $b_1, \ldots, b_8$ and a parity bit p, that is, consider the set of variables $V = \{b_1, \ldots, b_8, p\}$; the parity bit takes the value 1 if an even number of bits is one, otherwise it takes the value 0. The independency base L^I over V with these functional dependencies is $\{I(b_i, V \backslash b_i, \emptyset) | i = 1, \ldots, 8\} \cup \{I(p, V \backslash p, \emptyset)\}$. So, there are no independency statements in the associated independency model M^I_L that are not trivial, that is, for all independency statements

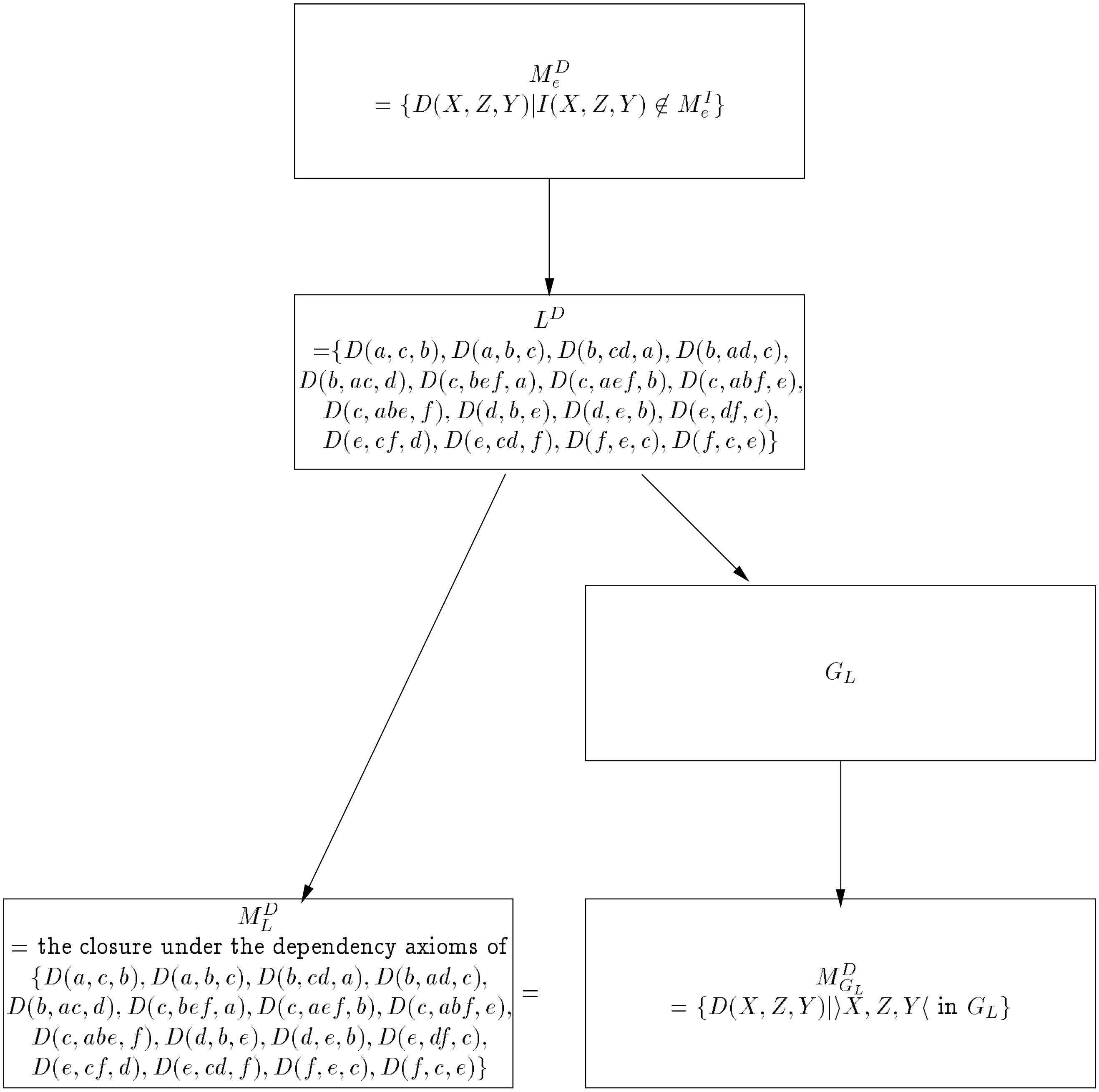

M_e^D
$= \{D(X,Z,Y) | I(X,Z,Y) \notin M_e^I\}$
L^D
$=\{D(a,c,b), D(a,b,c), D(b,cd,a), D(b,ad,c),$
$D(b,ac,d), D(c,bef,a), D(c,aef,b), D(c,abf,e),$
$D(c,abe,f), D(d,b,e), D(d,e,b), D(e,df,c),$
$D(e,cf,d), D(e,cd,f), D(f,e,c), D(f,c,e)\}$
G_L
M_L^D
= the closure under the dependency axioms of
$\{D(a,c,b), D(a,b,c), D(b,cd,a), D(b,ad,c),$
$D(b,ac,d), D(c,bef,a), D(c,aef,b), D(c,abf,e),$
$D(c,abe,f), D(d,b,e), D(d,e,b), D(e,df,c),$
$D(e,cf,d), D(e,cd,f), D(f,e,c), D(f,c,e)\}$
=
$M_{G_L}^D$
$= \{D(X,Z,Y) | \rangle X, Z, Y \langle$ in $G_L\}$

Figure 3.4: Relationship between dependency models, dependency base, undirected graph, and represented dependency model for M_e^I.

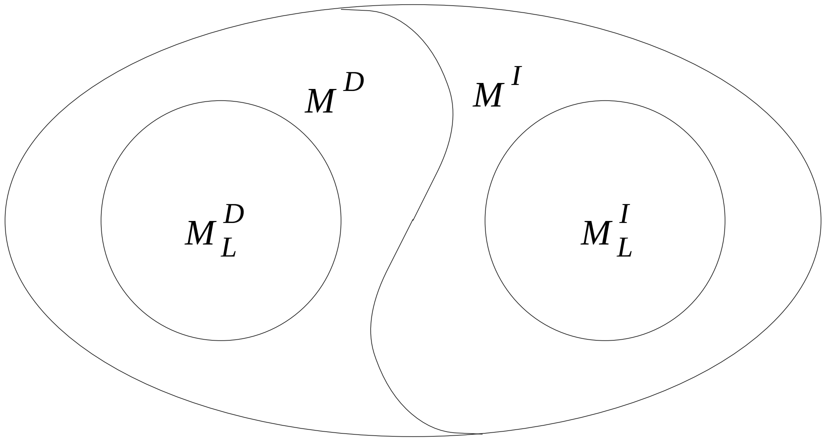


Figure 3.5: Division of the dependency pool

$I(X, Z, Y)$ in M_L^I at least one of X and Y is an empty set. The dependency base L^D constructed from L^I contains only dependency statements of the form $D(u, V\backslash uv, v)$. Now observe that only the symmetry and weak reunion axioms can be applied to these statements; the other dependency axioms cannot be used for deriving new dependency statements from L^D. So, the dependency model M_L^D associated with L^D contains only statements $D(X, Z, Y)$ for which $XYZ = V$ and M_L^D is relatively small. As a result, the set of statements in the dependency pool that cannot be identified as independency statements or dependency statements, that is, the set $M\backslash M_L^I M_L^D$, may be very large.

We showed that the consequences of a dependency base under the dependency axioms are valid dependency statements. Since the above example illustrates that the dependency base M_L^D associated with a dependency base L^D may be very small, one may ask whether extending L^D with more dependency statements would result in a larger M_L^D. These dependency statements should be easily obtainable from the independency base L^I, since we do not want to consult external sources of information about dependence. The following argument shows extending L^D is not useful. The only way to show that a dependency statements $D(X, Z, Y)$ is valid is by contradiction. We know of any neighborhood ν_u in L^I that no proper variable v of ν_v exists such that $I(u, \nu_u\backslash v, V\backslash uv\nu_u)$ is valid. The contradiction that needs to be derived is that under the assumption that $I(X, Z, Y)$ is valid, such a variable v does exist. The derivation uses the graphoid axioms. Now, using the dependency axioms, a complementary derivation can be constructed starting with $D(u, \nu_u\backslash v, v)$ and ending with $D(X, Z, Y)$. The dependency axiom used in the kth derivation step is the complement of the graphoid axiom used in the $n \Leftrightarrow k$th derivation with independency statements. So, every dependency statement that can be derived by contradiction using the independency base L^I and the independency model M_L^I can be derived from the dependency base. Therefore, it is not useful to extend the dependency base with other dependency statements if only knowledge of the independency base L^I may be used.

3.2.3 A Graphical Criterion for Conditional Dependence in Undirected Graphs

Recall from the theory on conditional independence that for a given independency base L^I, all statements in its associated independency model M_L^I can be read from the undirected graph G_L associated with L^I using the separation criterion. The notion of coupling offers a graphical criterion for dependency statements in M_L^D.

3.7 **Definition** Let V be a set of variables. Let $G = (V, E)$ be an undirected graph, and let X, Y, and Z be disjoint subsets of V. We call X and Y *coupled* given Z in G, written $\rangle X, Z, Y\langle_G$, if two nodes $u \in X$, $v \in Y$ or $u \in Y$, $v \in X$ exist such that

1. $u \Longleftrightarrow v$ is an edge in G, and
2. $\nu_u \subset XYZ$ where ν_u is the neighborhood of u in G. □

A statement $\rangle X, Z, Y\langle_G$ is called a *coupling statement*. In the sequel, we will omit the subscript G from coupling statements whenever the context makes it clear which graph G is considered.

To illustrate the notion of coupling, we consider the undirected graph of Figure 3.3. In this graph we have $\rangle a, c, b\langle$, because a and b are adjacent and the boundary of a, $\nu_a = bc$, is a subset of abc. Likewise, we have $\rangle ad, \emptyset, bce\langle$. In the graph, the statement $\rangle a, bd, e\langle$ does not hold, as a and e are not adjacent. Also the statement $\rangle a, \emptyset, c\langle$ does not hold in G as neither $\nu_a = bc$ nor $\nu_c = abef$ are subsets of ae.

Coupling has the property that if X and Y are coupled by Z in a minimal I-map, then X and Y are conditionally dependent given Z. So using coupling, one can read valid dependency statements from graphs that are minimal I-maps.

3.3 **Lemma** Let V be a set of variables. Let M^I be a graphoid independency model over V, and let $G_L = (V, E)$ be an undirected graph that is the minimal I-map of M^I. Let L^D be the dependency base of M^I. Let M_L^D be the dependency model associated with L^D. Then

$$\rangle X, Z, Y\langle_{G_L} \Rightarrow D(X, Z, Y) \in M_L^D,$$

for all disjoint subsets $X, Y, Z \subseteq V$. □

Proof: We have to show that if $\rangle X, Z, Y\langle_{G_L}$ then $D(X, Z, Y)$ can be derived from the dependency base L^D using the dependency axioms. Note that an undirected graph that is a minimal I-map of a graphoid independency model is unique and thus that G_L is the graph associated with the independency base L^I of M^I. Let $\rangle X, Z, Y\langle_{G_L}$ for some disjoint $X, Y, Z \subset V$. Without loss of generality, let $u \in X$ and $v \in Y$ such that u and v fulfill the properties mentioned in the definition of coupling. Since $u \Longleftrightarrow v$ in G_L, we know that $D(u, \nu_u \backslash v, v) \in L^D$. So, by Lemma 3.2 we have

$$D(u, \nu_u \backslash v, v) \in M_L^D.$$

Using the composition axiom with $Z \backslash \nu_u$ gives

$$D(u, \nu_u \backslash v, v(Z \backslash \nu_u)) \in M_L^D.$$

From L^I, we have that $I(u, \nu_u, V\backslash(u\nu_u)) \in M_L^I$. By using decomposition, we find $I(u, \nu_u, Z\backslash\nu_u) \in M_L^I$. Applying the intersection axiom with this gives

$$D(u, Z\nu_u\backslash v, v) \in M_L^D.$$

By composition and symmetry we obtain

$$D(X\backslash\nu_u, Z\nu_u\backslash v, v(Y\backslash\nu_u)) \in M_L^D.$$

We now apply weak reunion and symmetry and get

$$D(X, Z, Y) \in M_L^D.$$

□

In addition, coupling has the property that if X and Y are conditionally dependent given Z in M_L^D, then X and Y are coupled by Z in the minimal I-map G_L.

3.4 **Lemma** Let V be a set of variables. Let M^I be a graphoid independency model over V, and let $G_L = (V, E)$ be an undirected graph that is the minimal I-map of M^I. Let L^D be the dependency base of M^I. Let M_L^D be the dependency model associated with L^D. Then

$$D(X, Z, Y) \in M_L^D \Rightarrow \rangle X, Z, Y\langle_{G_L},$$

for all disjoint subsets $X, Y, Z \subseteq V$. □

Proof: Let $D(X, Z, Y) \in M_L^D$ be an arbitrary dependency statement for some disjoint sets $X, Y, Z \subseteq V$. It follows by definition that a derivation $\sigma_1 \overset{\gamma_1}{\Rightarrow} \sigma_2 \overset{\gamma_2}{\Rightarrow} \dots \overset{\gamma_k}{\Rightarrow} \sigma_{k+1}$ exists such that σ_i, $1 \leq i \leq k$, is a clause of the form $D(A, B, C)$ or $D(A, B, C) \wedge I(D, E, F)$, σ_{k+1} is $D(X, Z, Y)$, and γ_j, $1 \leq j \leq k$, is one of the dependency axioms. In the derivation, σ_1 is of the form $D(u, \nu_u\backslash v, v)$ where ν_u is the boundary of u in L^D. We prove the lemma by induction on the length k of the derivation of $D(X, Z, Y)$.

For $k = 0$, $D(X, Z, Y)$ is of the form $D(u, \nu_u\backslash v, v)$. Note that an undirected graph that is a minimal I-map of a graphoid independency model is unique and thus that G_L is the graph associated with the independency base L^I of M^I. By construction of G_L and definition of coupling, it is follows that $\rangle u, \nu_u\backslash v, v\langle_{G_L}$ holds.

Now, assume that $D(X, Z, Y) \in M_L^D \Rightarrow \rangle X, Z, Y\langle_{G_L}$ holds for all statements $D(X, Z, Y)$ derived from L^D in $k \Leftrightarrow 1$ steps. Then, a statement that can be derived in the kth step is constructed from a statement derived in $k \Leftrightarrow 1$ steps and one of the six dependency axioms. This leads to the following cases.

- Let γ_k be the symmetry ($D(X, Z, Y) \Leftrightarrow D(Y, Z, X)$), composition ($D(X, Z, Y) \Rightarrow D(X, Z, WY)$) or weak reunion axiom ($D(X, ZW, Y) \Rightarrow D(X, Z, WY)$). Then, the condition for coupling holds for the right-hand side of the axiom, if it holds for its left-hand side.

- Let γ_k be the extraction axiom ($D(X, Z, WY) \wedge I(X, Z, W) \Rightarrow D(X, ZW, Y)$). By the induction hypothesis $D(X, Z, WY) \in M^D$ implies that $\rangle X, Z, WY\langle_{G_L}$.

Then, the second property for the condition of coupling is obvious for the right-hand side $D(X, ZW, Y)$. We now consider the first property.

Let $u \in X$ and $v \in WY$ such that $u \Leftrightarrow v$ and $\nu_u \subset WXYZ$. Suppose that $v \in W$. Then $I(X, Z, W)$ cannot be represented by G_L since a node in X is adjacent to W, because of the edge $u \Leftrightarrow v$. Therefore, $v \in Y$ and the first property for the condition of coupling remains valid on the right-hand side. A similar argument applies when $u \in WY$ and $v \in X$ such $u \Leftrightarrow v$ and $\nu_v \subseteq WXYZ$.

- Let γ_k be the extraction+ axiom $(D(X, Z, WY) \wedge I(X, ZY, W) \Rightarrow D(X, Z, Y))$. By the induction hypothesis $D(X, Z, WY) \in M_L^D$ implies that $\rangle X, Z, WY \langle_{G_L}$.

  Let $u \in X$ and $v \in WY$ be such that $u \Leftrightarrow v$ and $\nu_u \subseteq WXYZ$. Assume $v \in W$. Then $I(X, ZY, W)$ cannot be represented by G_L. Therefore, $v \in Y$ and the first property for the condition of coupling remains valid. Assume a node $w \in W$ is in ν_u. Again, $I(X, ZY, W)$ cannot be represented by G_L. Therefore, no node $w \in \nu_u$ is in W and the second property for the condition of coupling remains valid for the right-hand side. A similar argument applies when $u \in WY$ and $v \in X$ exist such that $u \Leftrightarrow v$ and $\nu_v \subseteq WXYZ$.

- Let γ_k be the intersection axiom $(D(X, Z, WY) \wedge I(X, ZY, W) \Rightarrow D(X, ZW, Y))$. For this axiom, an argument analogous to that of extraction holds.

We conclude that for each dependency statement $D(X, Z, Y)$ that is derived in k steps from L^D by the dependency axioms implies that $\rangle X, Z, Y \langle_{G_L}$. This completes the induction and, hence, the proof of the lemma. □

From Lemmas 3.3 and 3.4 we conclude the following important property.

3.6 **Theorem** Let V be a set of variables. Let M^I be a graphoid independency model over V, and let $G = (V, E)$ be an undirected graph that is the minimal I-map of M^I. Let L^D be the dependency base of M^I. Let M_L^D be the dependency model associated with L^D. Then

$$\rangle X, Z, Y \langle_G \Leftrightarrow D(X, Z, Y) \in M_L^D,$$

for all disjoint subsets $X, Y, Z \subseteq V$. □

Separation and coupling are powerful criteria for reading independency and dependency statements from an undirected graph that is known to be a minimal I-map of some independency model. For stochastic independency models, the given axiomatic characterization of independency statements is not complete [106]. So, as the axiomatization of dependency statements is based on this axiomatization of independencies, it cannot be complete for stochastic dependency models either. It may be that more statements can be read from the structure of undirected graphs when extra restrictions, such as being closed under other axioms [81], are imposed on the dependency model.

3.3 Directed Acyclic Graphs

Another representation formalism for dependency and independency models is the directed acyclic graph. In this section, we will investigate directed acyclic graphs in the same way as we did for undirected graphs.

3.3.1 Conditional Independence in Directed Acyclic Graphs

In this section, we review the relationship between a graphoid independency model, its causal input list for a given ordering, and its associated directed acyclic graph.

Let M^I be an independency model over a set of variables V and let $<_V$ be a total ordering on V. The causal input list $L^I_{<_V}$ over M^I is a set of independency statements $L^I_{<_V} = \{I(u, \pi_u, V_u \backslash \pi_u) | u \in V\}$ where for each variable u, V_u is the set of all variables less than u according to $<_V$. The set π_u in $I(u, \pi_u, V_u \backslash \pi_u) \in L^I_{<_V}$ is the smallest subset of V_u such that $I(u, \pi_u, V_u \backslash \pi_u) \in M^I$. For graphoid independency models, the sets π_u are unique for a given ordering.

In practice, the ordering $<_V$ on the variables will be a 'causal ordering'. If a variable u has a causal influence on the value of variable v, then u will occur earlier in the ordering than v. For example, consider a medical domain with diseases and symptoms. As diseases cause symptoms, diseases will typically occur earlier in the ordering than their associated symptoms. Time is a good heuristic for detecting causal influences in a domain. If u takes a value only after v does, then u cannot be a cause of v. In this case u will be placed higher in the ordering than v. Identifying causal influences in a domain is typically a task for an expert in this domain. If the ordering $<_V$ can be interpreted as a causal ordering, then a set π_u in a causal input list $L^I_{<_V}$ can be interpreted as the set of all direct causes of the variable u. In fact, this property gives the causal input list its name. If u causes v and v causes w then u causes w. For an ordering $u < v < w$, u is not a direct cause of w, so u need not be in the parent set of w. However, for the ordering $u < w < v$, u is an element of the parent set of w. This example makes it clear that the ordering $<_V$ has a great influence on the resulting causal input list.

Given the independency model M^I and a total ordering $<_V$, the sets π_u in $L^I_{<_V}$ as described above can be constructed as follows. Initially, π_u is set to V_u for each variable $u \in V$. Then, M^I is consulted for every independency statement $I(u, V_u \backslash v, v)$ with $v \in V_u$. If $I(u, V_u \backslash v, v)$ is in M^I, then v is removed from the set π_u of u. This process is repeated for every variable $u \in V$. The algorithm is given in pseudo code below. In practical applications, a domain expert acts as an oracle to reveal this kind of information.

Consider the independency model M^I_e shown in Figure 3.1 and consider the ordering $<_V = a < b < c < d < e < f$. The set π_e of the variable e is constructed by starting with $\pi_e = abcd$. Since $I(e, bcd, a)$ is in M^I_e, a is removed from π_e. Likewise, the variable b is removed. Because $I(e, abd, c)$ and $I(e, abc, d)$ are not in M^I_e, it is concluded that $\pi_e = cd$. Applying this construction to all variables in V gives the causal input list $L^I_{<_V}$ listed in Figure 3.6.

The directed acyclic graph $G_{<_V}$ associated with a causal input list has an arc $u \rightarrow v$ from u to v if and only if u is an element of the parent set of v. The directed acyclic graph $G_{<_V}$ associated with the causal input list $L^I_{<_V}$ from Figure 3.6 is shown

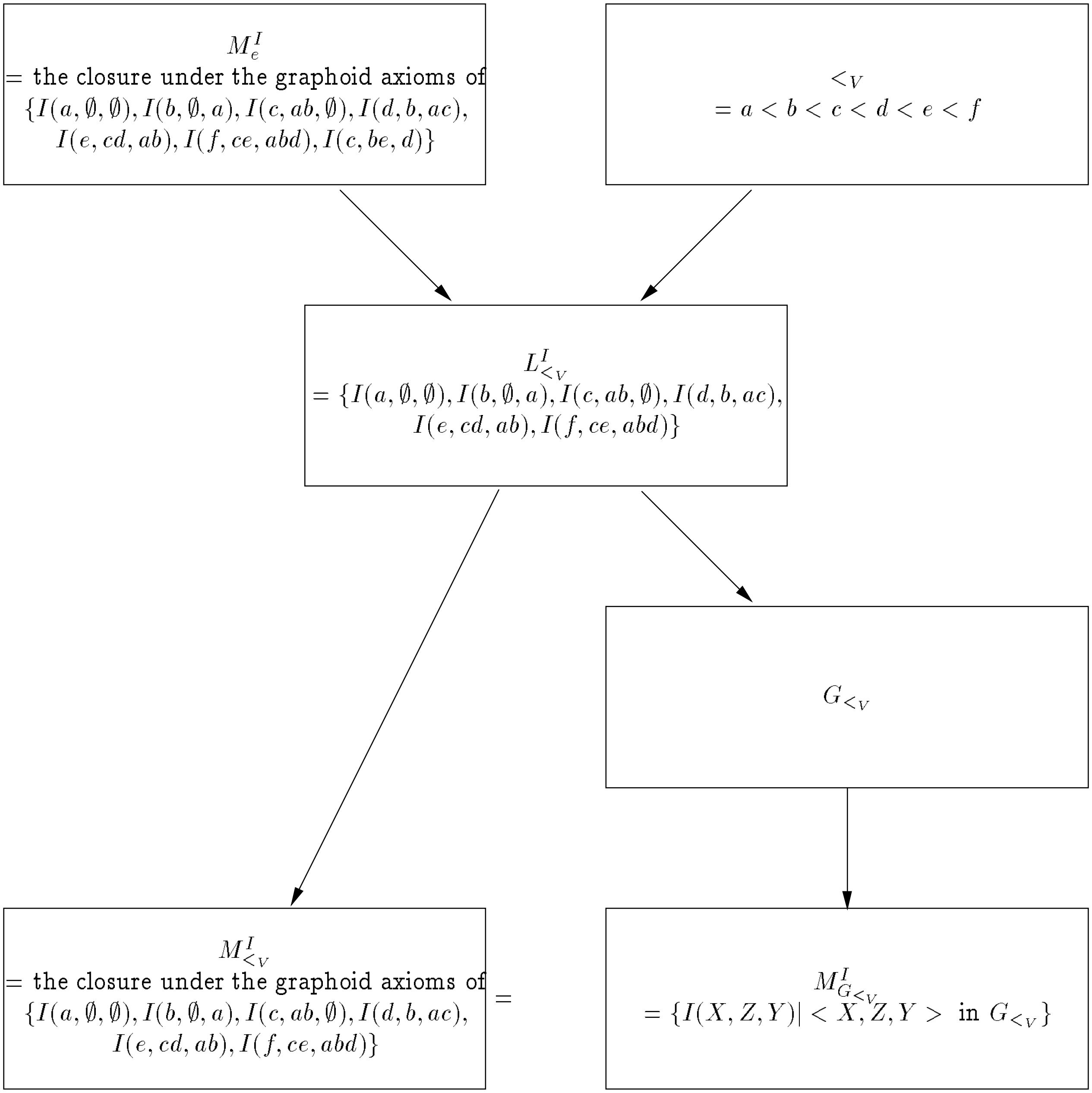

M_e^I
$=$ the closure under the graphoid axioms of
$\{I(a,\emptyset,\emptyset), I(b,\emptyset,a), I(c,ab,\emptyset), I(d,b,ac),$
$I(e,cd,ab), I(f,ce,abd), I(c,be,d)\}$
$<_V$
$= a < b < c < d < e < f$
$L_{<_V}^I$
$= \{I(a,\emptyset,\emptyset), I(b,\emptyset,a), I(c,ab,\emptyset), I(d,b,ac),$
$I(e,cd,ab), I(f,ce,abd)\}$
$G_{<_V}$
$M_{<_V}^I$
$=$ the closure under the graphoid axioms of
$\{I(a,\emptyset,\emptyset), I(b,\emptyset,a), I(c,ab,\emptyset), I(d,b,ac),$
$I(e,cd,ab), I(f,ce,abd)\}$
$=$
$M_{G_{<_V}}^I$
$= \{I(X,Z,Y) | < X,Z,Y > \text{ in } G_{<_V}\}$

Figure 3.6: Relationship between independency models, order, causal input list, directed acyclic graph, and represented independency model, with an example.

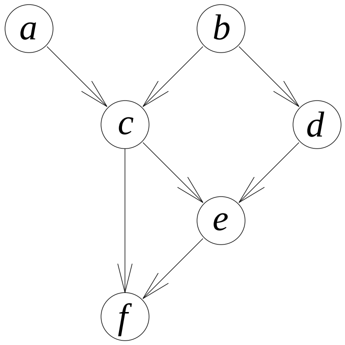


Figure 3.7: Directed acyclic graph induced by causal input list in Figure 3.6.

```
**Causal Input List Construction for** $M^I <_V$

for all $u \in V$ do  $\pi_u \leftarrow V_u$
for all $u \in V$ do
    for all $v \in V_u$ do
        if $I(u, V_v \backslash v, v) \in M^I$ then $\pi_u \leftarrow \pi_u \backslash v$
return $\{I(u, \pi_u, V_u \backslash \pi_u) | u \in V\}$
```

in Figure 3.7. For example, in $L^I_{<_V}$ we have that $\pi_e = cd$. So, in $G_{<_V}$ there is an arc $c \rightarrow e$ and an arc $d \rightarrow e$.

With the directed acyclic graph $G_{<_V}$, an independency model $M^I_{G_{<_V}}$ over V is associated containing for every $X, Z, Y \in V$ the independency statement $I(X, Z, Y)$ if and only if X and Y are separated by Z in G.

For example for the graph $G_{<_V}$ in Figure 3.7 we have that $\langle a, bc, d\rangle$: the chain a, c, e, d is blocked in $G_{<_V}$ since it satisfies the first condition of blocked chains (see Definition 2.22) because of node c; the chain a, c, b, d is blocked since $c \leftarrow b \rightarrow d$ satisfies the second condition; the chain a, c, f, e, d is blocked because the third condition is satisfied with node f. So, $M^I_{G_{<_V}}$ contains the independency statement $I(a, bc, d)$.

3.7 **Theorem** Let V be a set of variables, let $<_V$ be a total ordering on V, and let M^I be a graphoid independency model over V. Let $L^I_{<_V}$ be the causal input list over M^I and let $G_{<_V}$ be the directed acyclic graph associated with $L^I_{<_V}$. Then, $G_{<_V}$ is a minimal I-map of M^I. □

A proof of the theorem can be found in [82]. A well-known and useful property is that for a graphoid independency model M^I, the directed acyclic graph G that is a minimal I-map of M^I and obeys a total ordering, is unique. As a direct consequence of Theorem 3.7, we have the following property.

3.3 **Corollary** Let V be a set of variables, let $<_V$ be a total ordering on V, and let M^I be a graphoid independency model over V. Let $L^I_{<_V}$ be the causal input list over M^I and let $G_{<_V}$ be the directed acyclic graph associated with $L^I_{<_V}$. Then $M^I_{G_{<_V}} \subseteq M^I$. □

However, the converse of Theorem 3.7 does not hold. For example, $I(c, be, d)$ is in the independency model M_e^I listed in Figure 3.1 while it is not in $M_{G_{<_V}}^I$ with $<_V$ as before. The independency statements that are not represented by the directed acyclic graph in Figure 3.7 are marked with an asterisk in Figure 3.1. Only twenty-eight independency statements in the original independency model M_e^I are not represented by the directed acyclic graph. This illustrates the power for directed acyclic graphs of representing independency models.

A causal input list $L_{<_V}^I$ also induces an independency model $M_{<_V}^I$. This independency model $M_{<_V}^I$ is the closure of $L_{<_V}^I$ under the graphoid axioms.

For example, let $L_{<_V}^I$ be the causal input list in Figure 3.6. The independency statement $I(d, b, ac)$ is in $L_{<_V}^I$ so $I(d, b, ac) \in M_{<_V}^I$. Using weak union, we have $I(d, bc, a) \in M_{<_V}^I$. Note that $I(d, bc, a)$ also is element of the independency model $M_{G_{<_V}}^I$ associated with $L_{<_V}^I$. This is no coincidence, considering the following property.

3.8 **Theorem** Let V be set of variables. Let $<_V$ be an ordering on V. Let M^I be a graphoid independency model over V. Let $L_{<_V}^I$ be the causal input list over M^I and let $M_{<_V}^I$ be the independency model of $L_{<_V}^I$. Let $G_{<_V}$ be the directed acyclic graph associated with $L_{<_V}^I$ and let $M_{G_{<_V}}^I$ be the independency model of $G_{<_V}$. Then

$$M_{G_{<_V}}^I = M_{<_V}^I.$$

□

A proof is published in [82]. So, all independency statements that can be derived from a causal input list $L_{<_V}^I$ using the graphoid axioms are also represented in the graph induced by $L_{<_V}^I$ and vice versa.

The concepts described above can be illustrated using the independency model M_e^I listed in Figure 3.1. The relationship between the concepts is depicted in Figure 3.6. In the figure, a directed link between two rectangles means that the object at the tail of the arrow is sufficient to construct the object at the head of the arrow.

Note that the classes of independency models represented by undirected graphs and by directed acyclic graphs respectively are not equal. For example, the directed acyclic graph in Figure 3.7 represents $I(a, \emptyset, b)$ while no undirected graph exists that is an I-map of M_e^I in which $\langle a, \emptyset, b \rangle$. On the other hand, the undirected graph in Figure 3.3 represents both $I(b, cd, e)$ and $I(c, be, d)$ while no directed acyclic graph exists that represents both these statements and is an I-map of M_e^I.

3.3.2 Conditional Dependence in Directed Acyclic Graphs

In the previous subsection, it was shown that the representation of an independency model by a directed acyclic graph is equivalent to the representation by a causal input list. In this subsection, we will derive a similar property for dependency models.

3.8 **Definition** Let V be a set of variables. Let $<_V$ be a total ordering on V. Let M^I be an independency model over V and let $L_{<_V}^I$ be the causal input list over M^I. The *dependency base* $L^D_{<_V}$ of M^I is the set of dependency statements such that for each $u \in V$, $L^D_{<_V}$ contains dependency statements of the form

$$D(u, \pi_u \backslash v, v)$$

for each $v \in \pi_u$, where π_u is the parent set u such that $I(u, \pi_u, V_u \backslash \pi_u u) \in L^I_{<_V}$. □

Consider once more the independency model M^I_e listed in Figure 3.1 and the ordering $<_V = a < b < c < d < e < f$. For the variable c, the causal input list $L^I_{<_V}$ contains the independency statement $I(c, ab, \emptyset)$. So, $D(c, a, b)$ and $D(c, b, a)$ are in $L^D_{<_V}$. Note that the dependency base $L^D_{<_V}$ does not contain dependency statements of the form $D(a, X, Y)$ because $\pi_a = \emptyset$. Figure 3.8 shows the entire dependency base for M^I_e.

The motivation of the choice of the dependency base associated with a causal input list is the same as for dependency bases associated with an independency base. Namely, for minimal I-maps of a graphoid independency model M^I we know that the dependency statements $\{D(u, \pi_u \backslash v, v) | u \in V, v \in \pi_u\}$ are in the complement of M^I.

3.5 **Lemma** Let V be a set of variables. Let M^I be a graphoid independency model over V, and let M^D be its complementary graphoid dependency model. Let $<_V$ be an ordering on V. Let $L^D_{<_V}$ be the dependency base of M^I with $<_V$. Then

$$L^D_{<_V} \subseteq M^D.$$

□

Proof: The theorem will be proved by contradiction. Suppose that for some $u, v \in V$, $D(u, \pi_u \backslash v, v) \in L^D_{<_V}$ but $D(u, \pi_u \backslash v, v) \notin M^D$. From $D(u, \pi_u \backslash v, v) \notin M^D$ we have by definition that $I(u, \pi_u \backslash v, v) \in M^I$. By construction of $L^D_{<_V}$ we know that $I(u, \pi_u, V_u \backslash \pi_u) \in L^I_{<_V}$ and thus $I(u, \pi_u, V_u \backslash \pi_u) \in M^I$. From $I(u, \pi_u \backslash v, v) \in M^I$ and $I(u, \pi_u, V_u \backslash \pi_u) \in M^I$ we conclude that $I(u, \pi_u \backslash v, V_u \backslash (\pi_u \backslash v)) \in M^I$ using the contraction axiom. But then there is a proper subset S of π_u such that $I(u, S, V_u \backslash S) \in M^I$ and, hence, $L^I_{<_V}$ cannot be the causal input list of M^I. From the contradiction we conclude that $L^D_{<_V} \subseteq M^D$. □

The given proof is similar to that of Lemma 3.2. We can associate a dependency model with a dependency base $L^D_{<_V}$ in a similar way as we associated an independency model $M^I_{<_V}$ with a causal input list $L^I_{<_V}$.

3.9 **Definition** Let V be a set of variables. Let M^I be a graphoid independency model over V and let $<_V$ be an ordering on V. Let $L^I_{<_V}$ be the causal input list of M^I and let $M^I_{<_V}$ be the independency model associated with $L^I_{<_V}$. Let $L^D_{<_V}$ be the dependency base of M^I. The dependency model $M^D_{<_V}$ associated with L^D is the closure of L^D under the six dependency axioms, where the independency statements used are elements of $M^I_{<_V}$. □

Note that for determining $M^D_{<_V}$, derivations with independency statements may occur. Only independency statements contained in $M^I_{<_V}$ are used for these derivations. Note that not M^I is used as source of independency statements since M^I cannot be efficiently represented. Also note that, because the independency model $M^I_{G_{<_V}}$ is equal to $M^I_{<_V}$ (Theorem 3.8), these are exactly the independency statements that can be read from $G_{<_V}$ using the separation criterion.

No extra dependency statements need to be added to the dependency base $L^D_{<_V}$ to get a larger associated dependency model $M^D_{<_V}$, since all dependency statements that can possibly be derived given a causal input list of an unknown independency model M^I using the dependency axioms, can be derived from the dependency base.

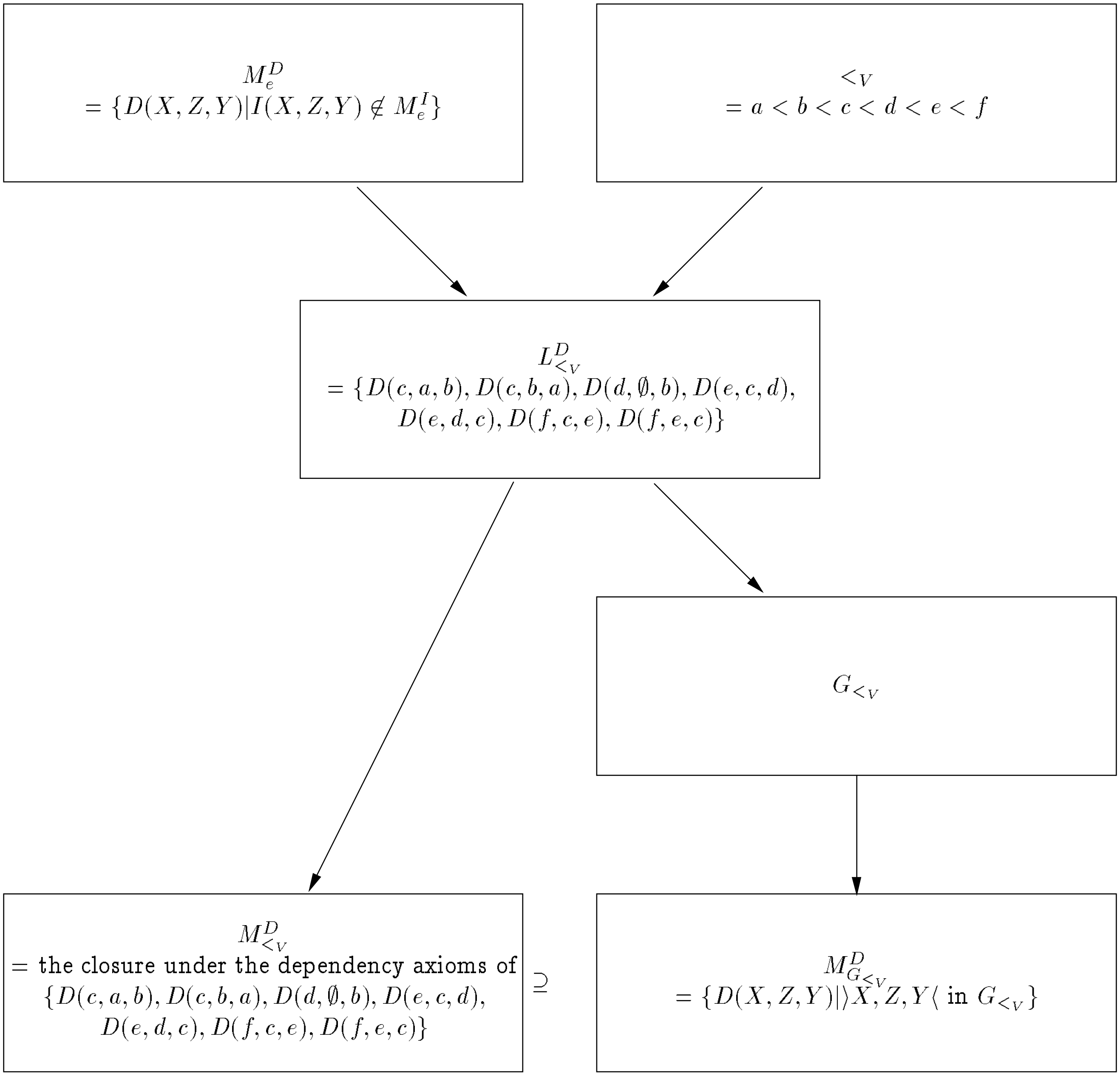

M^D_e
$= \{D(X,Z,Y) | I(X,Z,Y) \notin M^I_e\}$
$<_V$
$= a < b < c < d < e < f$
$L^D_{<_V}$
$= \{D(c,a,b), D(c,b,a), D(d,\emptyset,b), D(e,c,d),$
$D(e,d,c), D(f,c,e), D(f,e,c)\}$
$G_{<_V}$
$M^D_{<_V}$
= the closure under the dependency axioms of
$\{D(c,a,b), D(c,b,a), D(d,\emptyset,b), D(e,c,d),$
$D(e,d,c), D(f,c,e), D(f,e,c)\}$
$\supseteq$
$M^D_{G_{<_V}}$
$= \{D(X,Z,Y) | \rangle X,Z,Y \langle$ in $G_{<_V}\}$

Figure 3.8: Relationship between dependency models, dependency base, directed acyclic graph, and represented dependency model, with example.

3.9 **Theorem** Let V be a set of variables. Let M^I be a graphoid independency model over V and let M^D be its complementary graphoid dependency model. Let $<_V$ be a total ordering on V. Let $L^D_{<_V}$ be the dependency base of M^I and let $M^D_{<_V}$ be its associated dependency model. Then

$$M^D_{<_V} \subseteq M^D.$$

□

Proof: From Lemma 3.5, we have that $L^D_{<_V} \subseteq M^D$. From Theorem 3.2, we have that any graphoid dependency model is closed under the dependency axioms. The theorem now follows from the definition of $M^D_{<_V}$. □

So, all dependency statements derived from a dependency base are valid dependency statements.

3.3.3 A Graphical Criterion for Conditional Dependence in Directed Acyclic Graphs

In Section 3.2.1 we argued that all independency statements in the independency model $M^I_{<_V}$ associated with a causal input list $L^I_{<_V}$ can be read from the directed acyclic graph $G_{<_V}$ constructed from $L^I_{<_V}$ using the separation criterion. In this section we investigate a graphical criterion for reading dependency statements from the graph $G_{<_V}$. Again, the concept of coupling offers such a criterion.

3.10 **Definition** Let V be a set of variables. Let $G = (V, E)$ be a directed acyclic graph. Let X, Y and Z be disjoint subsets of V. We say that X and Y are *coupled* given Z in G, written $\rangle X, Z, Y\langle_G$, if two nodes $u \in X$, $v \in Y$ or $u \in Y$, $v \in X$ exist such that the following conditions hold:

1. $v \rightarrow u$ is an arc in G,
2. $\pi_u \subset XYZ$, where π_u is the parent set of u in G, and
3. a set Q with $Z \subseteq Q \subseteq XYZ \backslash uv$ exists such that $\langle u, Q, v\rangle$ in the directed acyclic graph $G' = (V, E')$ where $E' = E\backslash(v, u)$. □

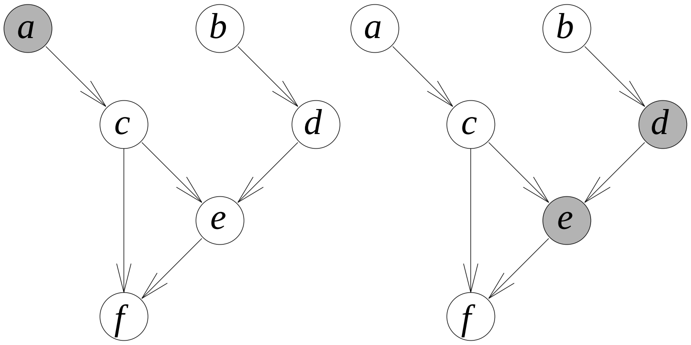


Figure 3.9: The nodes b and c are separated by a and by de.

A statement $\rangle X, Z, Y\langle_G$ is called a *coupling statement*. In the sequel, we will omit the subscript G from coupling statements as long as the context makes it clear which graph G is considered. The third property in the definition can be read as 'a set Q with $Z \subseteq Q \subseteq XYZ\backslash uv$ can be found such that u and v are d-separated by Q in the graph G', where G' is equal to G with the arc $v \rightarrow u$ removed.

To illustrate the notion of coupling, we consider the directed acyclic graph shown in Figure 3.7. In this directed graph, we have $\rangle c, a, b\langle$ since $b \rightarrow c$ is an arc in the graph, $\pi_c \subset abc$ and all chains between c and b not containing $b \rightarrow c$, that is, c, e, d, b and c, f, e, d, b, are blocked by $Q = a$ as the graph in Figure 3.9 on the left-hand side makes clear. Likewise we have $\rangle ac, e, bd\langle$ since $b \rightarrow c$ is an arc in the graph and $\pi_c \subset abcde$ and further $\langle b, de, c\rangle$ in the network structure when the arc $b \rightarrow c$ is removed, as Figure 3.9 on the right-hand side makes clear. For the graph, the statement $\rangle c, ae, b\langle$ does not hold since the chain c, e, d, b is not blocked by ae after removal of the arc $b \rightarrow c$. Similarly, the statement $\rangle c, de, b\langle$ does not hold since π_c is not a subset of $bcde$.

Coupling has the property that if X and Y are coupled by Z in a minimal I-map, then X and Y are conditionally dependent given Z.

3.6 **Lemma** Let V be a set of variables. Let M^I be a graphoid independency model over V. Let $<_V$ be a total ordering on V. Let $L^D_{<_V}$ be the dependency base of M^I and $<_V$ and let $G_{<_V}$ be its associated directed acyclic graph. Let $M^D_{<_V}$ be the dependency model associated with $L^D_{<_V}$. Then

$$\rangle X, Z, Y\langle_G \Rightarrow D(X, Z, Y) \in M^D_{<_V},$$

for all disjoint subsets $X, Y, Z \subseteq V$. □

Proof: We have to show that if $\rangle X, Z, Y\langle_{G_{<_V}}$ then $D(X, Z, Y) \in M^D_{<_V}$.

Let $\rangle X, Z, Y\langle_{G_{<_V}}$ for some $X, Y, Z \subset V$. Without loss of generality, let $u \in X$ and $v \in Y$ be nodes such that u and v fulfill the properties mentioned in the definition of coupling, that is, $v \rightarrow u$ is an arc in $G_{<_V}$, $\pi_u \subset XYZ$ and Q the set such that $Z \subseteq Q \subseteq XYZ\backslash uv$ and $\langle u, Q, v\rangle$ in $G_{<_V}$ when the arc $v \rightarrow u$ is removed.

From Lemma 3.5 we know that

$$D(u, \pi_u\backslash v, v) \in M^D_{<_V}.$$

Let $Q_d \subseteq Q$ be the set of nodes from Q that are descendants of u in $G_{<_V}$ and let $Q_a = Q\backslash Q_d\pi_u$. Applying the composition axiom, we find that

$$D(u, \pi_u\backslash v, vQ_a) \in M^D_{<_V}.$$

Now observe that by the global Markov property, we have $\langle u, \pi_u, V\backslash u\pi_u D_u\rangle$ where D_u is the set of descendants of u. This implies $I(u, \pi_u, V\backslash u\pi_u D_u) \in M^I_{G_{<_V}}$ by definition. Since $Q_a \subset V\backslash u\pi_u D_u$, we can apply the decomposition axiom to get $I(u, \pi_u, Q_a) \in M^I_{G_{<_V}}$. By applying intersection with $D(u, \pi_u\backslash v, vQ_a) \in M^D_{<_V}$ we get

3.1

$$D(u, Q_a\pi_u\backslash v, v) \in M^D_{<_V}.$$

Now we prove that $D(u, Q\pi_u\backslash v, v) \in M^D_{<_V}$ by successively adding nodes from Q_d to $Q_a\pi_u\backslash v$, using Lemma 3.8 which can be found in the appendix. If the arc $v \rightarrow u$ is deleted from $G_{<_V}$, all chains in the modified graph between u and v blocked by

$Q_a \backslash v$ are also blocked by $Q_a\pi_u\backslash v$. Let $z \in Q_d$ be the node that is minimum in $<_V$, that is, for all $v \in Q_d\backslash z$, $z <_V v$. Then we can use Lemma 3.8 and know that either $I(u, Q_a\pi_u, z)$ or $I(v, uQ_a\pi_u\backslash v, z)$. So, either we apply composition with z on Formula 3.1 and we get

$$D(u, Q_a\pi_u\backslash v, vz) \in M^D_{<_V}.$$

Then, intersection with $I(u, Q_a\pi_u, z)$ gives

$$D(u, zQ_a\pi_u\backslash v, v) \in M^D_{<_V}.$$

Or, we apply symmetry first on Formula 3.1 and then composition with z, and we get

$$D(v, Q_a\pi_u\backslash v, uz) \in M^D_{<_V}.$$

Now, intersection with $I(v, uQ_a\pi_u\backslash v, z)$ again gives

$$D(v, zQ_a\pi_u\backslash v, u) \in M^D_{<_V}.$$

Repeat this derivation with $Q_a = Q_a \cup z$ and $Q_d = Q_d\backslash z$ until $Q_d = \emptyset$. Observe that if all chains between u and v not containing $v \to u$ are blocked by a set D then they are also blocked by a set $D\backslash p$ where p is a descendant of u that has no descendants in D. Therefore, Lemma 3.8 applies every time the derivation is repeated.

This results eventually in $D(u, Q\pi_u\backslash v, v)$. So,

$$D(u, Q\pi_u\backslash v, v) \in M^D_{<_V}.$$

To this dependency statement we apply the composition axiom and the symmetry axiom, to get

$$D(u(X\backslash Q\pi_u), Q\pi_u\backslash v, v(Y\backslash Q\pi_u)) \in M^D_{<_V}.$$

Using weak reunion and symmetry we now find

$$D(X, Z, Y) \in M^D_{<_V}.$$

□

The lemma implies that coupling provides a sufficient condition for reading dependency statements from a minimal I-map. However, it does not provide a necessary condition. For example, consider once more the directed acyclic graph from Figure 3.7 which is a minimal I-map of the independency model M^I_e. We know from the dependency base that $D(c, a, b) \in M^D_{<_V}$. So, by the symmetry axiom and the composition axiom, we find $D(b, a, cef) \in M^D_{<_V}$. From the graph we read that $I(b, ace, f) \in M^I_{<_V}$ using the separation criterion. $D(b, a, cef) \in M^D_{<_V}$ and $I(b, ace, f) \in M^I_{<_V}$ now imply $D(b, af, ce) \in M^D_{<_V}$ using the intersection axiom. However, the coupling statement $\rangle b, af, ce\langle$ does not hold for the graph $G_{<_V}$ from Figure 3.7.

Since $M^D_{<_V} \subseteq M^D$, we have the following theorem as a consequence of Lemma 3.6.

3.10 **Theorem** Let V be a set of variables. Let M^I be a graphoid independency model over V, and let M^D be its complementary dependency model. Let G be a directed acyclic graph that is a minimal I-map of M^I. Then

$$\rangle X, Z, Y \langle_G \Rightarrow D(X, Z, Y) \in M^D,$$

for all disjoint subsets $X, Y, Z \subseteq V$. □

Proof: Let $<_V$ be an ordering on V such that G associated with the dependency base $L^D_{<_V}$ of M^I is equal to G. Note that any topological ordering on the nodes in G is such an ordering. Let $L^D_{<_V}$ be the dependency base of M^I and $<_V$ and let $M^D_{<_V}$ be the dependency model associated with $L^D_{<_V}$. Then, by Lemma 3.6 we have $\rangle X, Z, Y \langle_G \Rightarrow D(X, Z, Y) \in M^D_{<_V}$. The theorem now follows from the fact that $M^D_{<_V} \subseteq M^D$. □

The question arises whether a mathematically appealing and not too complex graphical criterion exists that provides both a sufficient and a necessary condition to read all dependency statements in $M^D_{<_V}$ from a minimal I-map $G_{<_V}$. We do not believe that such a criterion exists because of the following reason. Consider a directed acyclic graph $G_{<_V}$ constructed from a causal input list $L^I_{<_V}$ over an independency model M^I. Let $D(X, Z, Y)$ be a dependency statement in $M^D_{<_V}$. Then, a derivation exists starting with $D(u, \pi_u \backslash v, v)$ and ending with $D(X, Z, Y)$. By structural induction it can be shown that in every step of this derivation the following property is preserved. If $D(X', Z', Y')$ is the result of one step in the derivation, then two nodes $u \in X'$ and $v \in Y'$ or $u \in Y'$ and $v \in X'$ exist such that $v \rightarrow u$ and $\pi_u \subset XYZ$. So, any graphical criterion for reading dependency statements from a minimal I-map must satisfy this property.

3.7 **Lemma** Let V be a set of variables. Let M^I be a graphoid independency model over V and let M^D be its complementary graphoid dependency model. Let $<_V$ be a total ordering on V. Let $L^D_{<_V}$ be the dependency base of M^I and $<_V$ and let $G_{<_V}$ be its associated directed acyclic graph. Let $M^D_{<_V}$ be the dependency model associated with $L^D_{<_V}$. Then, a necessary condition for a graphical criterion that holds for any triple (X, Z, Y) where X, Y, and Z are disjoint subsets of V if and only if $D(X, Z, Y) \in M^D_{<_V}$ is the following: there exist two nodes $u \in X$, $v \in Y$ or $u \in Y$, $v \in X$ such that

1. $v \rightarrow u$ is an arc in $G_{<_V}$, and
2. $\pi_u \subset XYZ$ in $G_{<_V}$. □

Proof: Let $D(X, Z, Y) \in M^D_{<_V}$ for some disjoint sets $X, Y, Z \in V$. It follows by definition that a derivation $\sigma_1 \overset{\gamma_1}{\Rightarrow} \sigma_2 \overset{\gamma_2}{\Rightarrow} \ldots \overset{\gamma_k}{\Rightarrow} \sigma_{k+1}$ exists such that σ_i, $1 \leq i \leq k$, is a clause of the form $D(A, B, C)$ or $D(A, B, C) \wedge I(D, E, F)$, σ_{k+1} is $D(X, Z, Y)$, and γ_j, $1 \leq j \leq k$, is one of the dependency axioms. In this derivation, σ_1 is of the form $D(u, \pi_u \backslash v, v) \in L^D_{<_V}$. We prove the lemma by induction on the length k of the derivation of $D(X, Z, Y)$.

For $k = 0$, $D(X, Z, Y)$ is of the form $D(u, \pi_u \backslash v, v)$. It is easily verified that $v \rightarrow u$ is an arc in $G_{<_V}$ and $\pi_u \subseteq XYZ$.

Now assume that the lemma holds for all statements that can be derived in less than k steps for some $k \geq 1$. Then, a statement $D(X, Z, Y)$ that can be derived in the kth step is constructed from a statement derived in $k \Leftrightarrow 1$ steps and γ_k is one of the six dependency axioms. This leads to the following cases.

- Let γ_k be the symmetry axiom $(D(X, Z, Y) \Rightarrow D(Y, Z, X))$. By the induction hypothesis, we may assume without loss of generality that there are nodes $u \in X$ and $v \in Y$ such that $v \rightarrow u$ is an arc in $G_{<_V}$ and $\pi_u \subset XYZ$. In the statement $\sigma_{k+1} = D(Y, Z, X)$ we have that $u \in Y$, $v \in X$ such that $v \rightarrow u$ is an arc in $G_{<_V}$ and $\pi_u \subset XYZ$. So if γ_k is the symmetry axiom, the property stated in the lemma holds for σ_k.

  Similar observations hold if γ_k is the composition or weak reunion

- Let γ_k be the extraction+ axiom $(D(X, Z, WY) \wedge I(X, ZY, W) \Rightarrow D(X, Z, Y))$. By the induction hypothesis, the lemma holds for $D(X, Z, WY)$. Now, let $u \in X$ and $v \in WY$ such that $v \rightarrow u$ is an arc in $G_{<_V}$ and $\pi_u \subset WXYZ$. Suppose that $v \in W$. Then, $\langle X, ZY, W\rangle$ does not hold in $G_{<_V}$ since $v \rightarrow u$ in $G_{<_V}$ implies that X and W are not separated in $G_{<_V}$. But then, $I(X, ZY, W)$ cannot have been used in the kth step of the derivation. From this contradiction, we conclude that v must be in Y. Now, consider $\sigma_{k+1} = D(X, Z, Y)$. It will be evident that there are two nodes $u \in X$ and $v \in Y$ such that $v \rightarrow u$ is an arc in $G_{<_V}$.

  It remains to be shown that for these nodes $\pi_u \subset XYZ$ holds. Suppose that a node $w \in \pi_u$ exists such that $w \in W$. Then, $\langle X, ZY, W\rangle$ does not hold in $G_{<_V}$. But then, $I(X, ZY, W)$ can not have been used in the kth step of the derivation. From the contradiction, we conclude that $\pi_u \subset XYZ$.

  Now consider the case where $u \in WY$ and $v \in X$ such that $v \rightarrow u$ is an arc in $G_{<_V}$ and $\pi_u \subset WXYZ$. Using a similar argument as before, we find that $u \notin W$ and hence $u \in Y$.

  Suppose that a node $w \in \pi_u$ exists such that $w \in W$. Then, there is a chain $v \rightarrow u \leftarrow w$ in $G_{<_V}$. So, $\langle X, ZY, W\rangle$ does not hold. But then, $I(X, ZY, W)$ can not have been used in the kth step of the derivation. From the contradiction we conclude that $\pi_u \subset XYZ$.

- Let γ_k be the extraction axiom $(D(X, Z, WY) \wedge I(X, Z, W) \Rightarrow D(X, ZW, Y))$ or the intersection axiom$(D(X, Z, WY) \wedge I(X, ZY, W) \Rightarrow D(X, ZW, Y))$. For these axioms, an argument analogous to that for extraction holds.

We conclude that the lemma holds for every dependency statement $D(X, Z, Y)$ that is derived in k steps from $L^D_{<_V}$ by the dependency axioms. □

Obviously, the coupling criterion given in Definition 3.10 satisfies these two conditions. Furthermore, a criterion that identifies all dependency statements must be able to identify the following dependency statements. As we saw in the previous example, $D(b, af, ce)$ is in $M^D_{<_V}$ of Figure 3.8. However, $D(be, af, c)$ can not be derived using the dependency axioms and the independency statements in the directed acyclic graph. This example shows that the derivation depends on the place of the variable e in the statement. As a consequence, we can have that $D(u, Z, vY)$ is an element of $M^D_{<_V}$ while $D(uY, Z, v)$ is not. A graphical criterion must reckon with this possibility.

For dependency statements $D(X, Z, Y)$ where X and Y consist of single variables this problem does not arise. We have the following result.

3.11 **Theorem** Let V be a set of variables. Let M^I be a graphoid independency model over V, and let M^D be its complementary graphoid dependency model. Let $<_V$ be an ordering on V. Let $L^D_{<_V}$ be the dependency base of M^I and $<_V$ and $G_{<_V}$ the associated directed acyclic graph. Let $M^D_{<_V}$ be the dependency model associated with $L^D_{<_V}$. Then,

$$\rangle u, Z, v \langle_{G_{<_V}} \Leftrightarrow D(u, Z, v) \in M^D_{<_V},$$

for every $u, v \in V$ and $Z \subseteq V$. □

Proof: The property $\rangle u, Z, v \langle_{G_{<_V}} \Rightarrow D(u, Z, v) \in M^D_{<_V}$ for all $u, v \in V$, $Z \subseteq V \backslash uv$ follows immediately from Theorem 3.10.

We prove $D(u, Z, v) \in M^D_{<_V} \Rightarrow \rangle u, Z, v \langle_{G_{<_V}}$ by contradiction. Suppose that there are two nodes $u, v \in V$ and a set $Z \subseteq V \backslash uv$ such that $D(u, Z, v) \in M^D_{<_V}$ and not $\rangle u, Z, v \langle_{G_{<_V}}$.

By Lemma 3.8, we know that either $u \rightarrow v$ and $\pi_u \subset vZ$ in $G_{<_V}$ or $v \rightarrow u$ and $\pi_v \subset uZ$ in $G_{<_V}$. Without loss of generality, we assume the former. Since u and v are not coupled by Z in $G_{<_V}$, we conclude that a chain $p = u, \ldots, v$ must exist that does not contain $v \rightarrow u$ and that is not blocked by any set Q such that $Z \subseteq Q \subseteq uvZ \backslash uv$, that is, by Z.

Since p is not blocked by Z and $\pi_u \subseteq vZ$, p cannot contain an arc to u from one of its parents. So, p must contain an arc from u to one of its children. The chain p therefore contains a v-node c. Since p is not blocked by Z, we have that Z contains c or one of its descendants. Let $w \in Z$ be such a node (such that length of the path from c to w is minimal).

Now consider the statement $D(u, Z, v)$. From $D(u, Z, v) \in M^D_{<_V}$. It follows by definition that a derivation $\sigma_1 \overset{\gamma_1}{\Rightarrow} \sigma_2 \overset{\gamma_2}{\Rightarrow} \ldots \overset{\gamma_k}{\Rightarrow} \sigma_{k+1}$ exists such that σ_i, $1 \leq i \leq k$, is a clause of the form $D(A_i, B_i, C_i)$ or $D(A_i, B_i, C_i) \wedge I(D_i, E_i, F_i)$, σ_{k+1} is $D(u, Z, v)$, and γ_j, $1 \leq j \leq k$, is one of the dependency axioms. In this derivation, σ_1 is of the form $D(u, \pi_u \backslash v, v) \in L^D_{<_V}$. By induction, it can be easily shown that for all dependency statements $D(A_i, C_i, B_i)$ derived from $D(u', \pi_{u'} \backslash v', v')$ we have either $u' \in A$ and $v' \in B$ or $u' \in B$ and $v' \in A$. We conclude that $D(u, Z, v)$ is derived from $\sigma_1 = D(u, \pi_u \backslash v, v)$.

On the other hand, if a derivation $\sigma_1 \overset{\gamma_1}{\Rightarrow} \sigma_2 \overset{\gamma_2}{\Rightarrow} \ldots \overset{\gamma_k}{\Rightarrow} \sigma_{k+1}$ exists, then also a derivation $\sigma'_{k+1} \overset{\gamma'_k}{\Rightarrow} \sigma'_k \overset{\gamma'_{k-1}}{\Rightarrow} \ldots \overset{\gamma'_1}{\Rightarrow} \sigma'_1$ exists where σ'_i, $1 \leq i \leq k+1$, is $I(A, B, C)$ if σ_i is $D(A_i, B_i, C_i)$ and $I(A_i, B_i, C_i) \wedge I(D_i, E_i, F_i)$ if σ_i is $D(A_i, B_i, C_i) \wedge I(D_i, E_i, F_i)$. So, $\sigma'_{k+1} = I(u, Z, v)$, $\sigma'_1 = I(u, \pi_u \backslash v, v)$. Further, γ'_j, $1 \leq j \leq k$ is the graphoid axiom corresponding to the dependency axiom γ_j.

To make this last derivation, the variable w, which is in $Z \backslash (\pi_u \backslash v)$, must be removed from a clause $\sigma_i = I(A_i, C_i, B_i)$. The only axiom with which this can be be performed is decomposition. But for decomposition, w must be in B_i. However, in σ'_{k+1} we have that w is in C_{k+1}. The only axioms with which the variable w can be move from C_i to B_{i+1} is contraction with an independency statements of the form $I(A_i, C_i\backslash w, w)$ where either u or v in A_i or intersection with an independency statements of the form $I(A_i, C_iB_i\backslash w, w)$.

However, regarding trail p in $G_{<_V}$, we have that if $C_i \subseteq Z$, no statement $\langle A_i, C_i\backslash w, w\rangle$ can hold in $G_{<_V}$ (note that introducing nodes to C_i that are not in Z would give rise to the same problem that we want to solve). Therefore, we conclude that $D(u, Z, v)$ cannot be derived unless $\rangle u, Z, v\langle$.

□

This theorem says that at least all dependency statements in $M^D_{<_V}$ concerning single nodes can be read from the graph. So, for this class of dependency statements, the coupling criterion provides both a sufficient and a necessary condition. These are exactly the kind of dependency statements we may be interested in when evaluating a network structure.

In summary, we defined a graphical criterion for reading dependency statements from directed acyclic graphs that are minimal I-maps. This criterion does not identify all dependency statements one may possibly know, given the information that the graph is a minimal I-map. However, all statements in the interesting class of dependency statements of the form $D(u, Z, v)$ that can be known, can be identified by the criterion.

3.4 Appendix

For the proof of Theorem 3.10 we use the following lemma.

3.8 **Lemma** Let V be a set of variables. Let $G = (V, E)$ be a directed acyclic graph. Let $u, v \in V$ be two nodes such that $v \to u$ in G. Let $Z \subseteq V\backslash uv$ be a set of nodes such that $\pi_u \subset Zv$. Let $c \in V\backslash Z$ be a node that is a descendant of node u and has no descendants in Z. Furthermore, let every chain between u and v not containing $v \to u$ be blocked by Zc. Then, either $\langle v, Zu, c\rangle$ or $\langle u, Zv, c\rangle$. □

Proof: We prove the lemma by contradiction.

Suppose that $p = u, \ldots, c$ is a chain in G such that p is not blocked by Zv (see right-hand side of Figure 3.10). Now suppose that p is blocked by Z. Then, p contains a v-node w such that v is a descendant of w and the chain $u, \ldots, w$ is not blocked by Z. But then, a chain $u, \ldots, w, \ldots, v$ not containing $v \to u$ exists in G that is not blocked by Zc. From the contradiction we conclude that p is not blocked by Z (see center picture of Figure 3.10).

Now suppose that $q = v, \ldots, c$ is a chain in G such that q is not blocked by Zu. Suppose that q is blocked by Z. Then, q contains a v-node w such that u is a descendant of w. For node u, we have $\pi_u \subset Zv$. So, a node $u \in Zv$ would exist that is a descendant of w and q would not be blocked by Zc either. So, if q is not blocked by Zu, then q is also not blocked by Z (see left-hand side of Figure 3.10).

Since c does not have any descendants in Z both chains p and q have an incoming arrow into c. It follows that the chain r composed of p and q is not blocked by Zc. But then, there is a chain $u, \ldots, c, \ldots, v$ not containing $v \to u$ that is not blocked

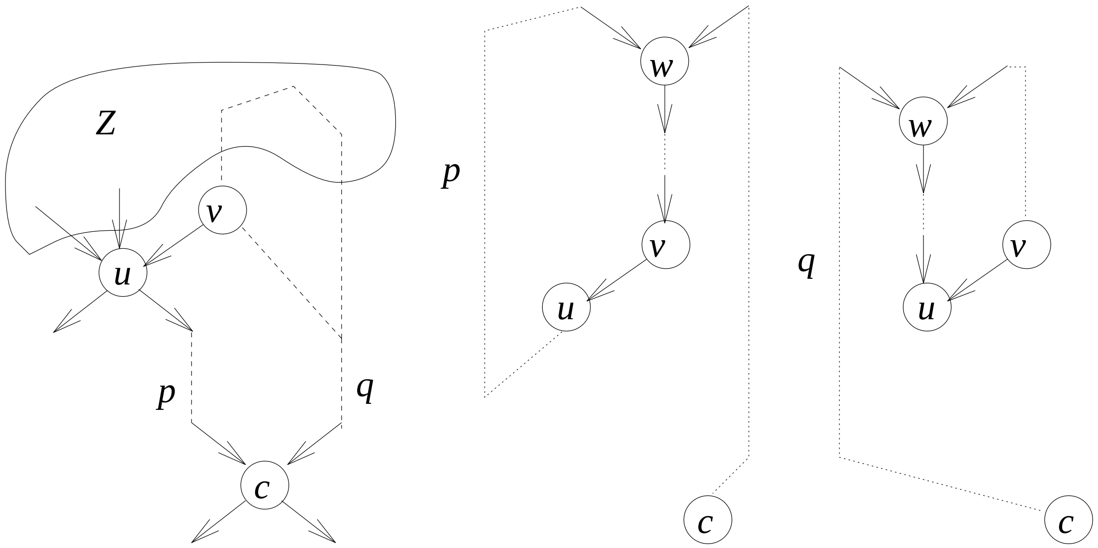


Figure 3.10: Possible chains in G.

by Zc. From the contradiction, we conclude that either all chains between u and c are blocked by Zv or all chains between v and c blocked by Zu. □

Learning Bayesian Networks

The task of constructing a Bayesian belief network is twofold: constructing the network structure and defining the set of assessment functions.

Construction of a Bayesian belief network with domain experts is a difficult and time-consuming task. In many domains are ill-understood. As a result, experts have problems in making causal relations explicit troubling the construction of network structures. It is this kind of domains where Bayesian belief networks are likely to be applied. In other domains where Bayesian belief networks are likely to be applied, there are few experts if any. Even if the causal relations in the domain can be made available easily, the definition of the assessment functions remains a time consuming task; human experts are good in judging qualitative relations but they are not good in quantizing these relations [112]. Therefore, the build-test cycle may need to be performed many times.

By exploiting information contained in databases, the construction time of Bayesian belief networks may be considerably decreased. An automatically constructed Bayesian belief network may be used directly for inference or considered as a starting point for the build-test cycle. In addition, the structure of an automatically constructed Bayesian belief network may give insight in the dependencies among the variables in the domain.

In learning a network structure, several network structures are generated and compared to decide which one is the ‘best’ structure, given the database. For the purpose of comparing structures a quality measure is used: the better the network fits the database, the higher the quality. The basic idea is that a network structure with a higher quality is preferred over a network structure with lower quality. In the literature, several quality measures have been proposed, building on different theories. In Section 4.2, we will investigate properties of the most popular quality measures.

For constructing network structures, a search algorithm is employed that selects network structures that are likely to be of high quality from the space of all possible structures. As the number of network structures grows exponentially in the number of variables, searching for a network structure with the highest quality may be a time-consuming task. Therefore, generally heuristics are applied. In Section 4.3, the complexity of this task will be investigated and several search algorithms are considered.

Once a network structure with high quality has been selected, the assessment functions for the structure can be estimated. This can be performed by direct estimation from the database. The accuracy of these estimates indicates how well the represented probability distribution captures the ‘real’ probability distribution. In most real-life applications, databases are relatively small compared to the number of probabilities that have to be estimated and a large error in the assessment functions can be expected. In Section 4.4, we will explore a technique that utilizes data more efficiently.

The theory developed in this chapter has been extensively tested in experiments with synthetical databases. The results of these experiments are presented in Section

4.5. However, we shall first list our basic assumptions in the following section.

4.1 Basic Assumptions

For learning a Bayesian belief network from data a database is used. We introduce the concept of a database and give some notational conventions.

4.1 Definition Let $V = \{v_1, \ldots, v_n\}$ be a set of variables. A *database* D over V is a multi-set of configurations of V. The configurations of V comprised in the database D are referred to as *cases*. □

In the definition of a database, there are several implicit assumptions. First of all, the variables involved are discrete: a database consists of configurations and configurations are defined for discrete variables only. Very little is known about quality measures for networks and databases over continuous variables. The most widely used technique to deal with databases with continuous variables is to transform the continuous variables into discrete variables. An alternative approach, based on the assumption that the underlying distribution is multivariate normal, is given by Geiger and Heckerman [43].

Another assumption is that there are no cases in a database for which there are variables for which a no value is specified. This assumption certainly does not hold for most real-world databases. The basic techniques for handling missing values is to treat a missing value of a variable as if it were an extra value for the variable, or to estimate a distribution over the variable's values to fill in the missing values in the database. Spiegelhalter et al. [101] give a survey of methods to circumvent this assumption.

Also, since a database is a multi-set of cases, no ordering exists on the cases in the database and all cases are independent of each other, given the model. So, it is assumed that the process that generated the data is time-independent. Methods that assume the existence of time dependence label the cases with a weighing factor that is smaller as the cases are obtained a longer time ago [76, 102].

Note that we allow databases with an infinite number of cases.

In this chapter, we will use a shortened notation. Let v_i be a variable in V. We write π_i instead of π_{v_i} to denote the parent set of v_i and r_i instead of r_{v_i} to denote the cardinality of Ω_{v_i}. We use q_i to denote $\prod_{u_j \in \pi_i} r_j$. Note that q_i equals the number of configurations of the parent set π_i which is one if $\pi_i = \emptyset$. The elements of Ω_{v_i} and Ω_{π_i} will be ordered $x_{i1}, \ldots, x_{ir_i}$ and $x_{\pi_i 1}, \ldots, x_{\pi_i q_i}$, respectively. Now, let x_{ik} denote the kth value of the variable v_i and $x_{\pi_i j}$ the jth configuration of the parent set π_i of v_i. Then, we write N to denote the cardinality of the database D, we write N_{ijk} to denote the number of cases in D where $v_i = x_{ik}$ and $\pi_i = x_{\pi_i j}$, and we write N_{ij} to denote the number of cases where $\pi_i = x_{\pi_i j}$.

4.2 Quality Measures

Quality measures can be built on various different approaches. In the sections 4.2.1 up to 4.2.3, we review the quality measures built on a Bayesian approach, an information criterion approach, and a minimum description length approach, respectively.

We compare the three measures in Section 4.2.4. In Section 4.2.5 we study whether these measures assign the same quality to equivalent network structures. The section is concluded with an investigation of the properties of the various quality measures for infinite-size databases in Section 4.2.6 and for finite-size databases in Section 4.2.7. But to fix the term quality measure, we will first give its definition.

4.2 **Definition** Let V be a set of variables, and let $\mathcal{D}$ be the set of all possible databases over V. Let $\mathcal{B}_S$ be the set of all possible network structures over V. Then, a *quality measure* $Q : \mathcal{B}_S \times \mathcal{D} \to (\Leftrightarrow\infty, 0]$ is a non-positive real-valued function. □

4.2.1 A Bayesian Approach

The Bayesian approach is a well-founded and practical method for selecting statistical models given a database. In the context of Bayesian belief network learning, the statistical model is a network structure. The basic idea of this approach is to use the posterior probability of a network structure given the database as a measure of the quality of the structure. These posterior probabilities are computed as follows. First, a prior distribution over all network structures is defined. Then, for each structure, the probability of the database given the structure is computed. Using Bayes' theorem, the posterior probability of the structure given the database is calculated. In order to compare the posterior probabilities of two network structures B_{S_1} and B_{S_2} we can calculate $P(B_{S_1}|D)/P(B_{S_2}|D)^1$ by

$$\frac{P(B_{S_1}|D)}{P(B_{S_2}|D)} = \frac{\frac{P(B_{S_1},D)}{P(D)}}{\frac{P(B_{S_2},D)}{P(D)}} = \frac{P(B_{S_1},D)}{P(B_{S_2},D)}.$$

So, it suffices to calculate $P(B_S, D)$ for all network structures B_S.

Cooper and Herskovits [24] have proposed a quality measure based on the Bayesian approach. For this purpose, they have derived a formula for computing the probabilities $P(B_S, D)$, based on the assumption that no set of assessment functions B_P is preferred for a given network structure before the database has been inspected. This assumption implies that the prior probability distribution over the values of the assessment functions is uniform. We will return to this assumption in Section 4.2.5. They derive the following formula for $P(B_S, D)$:

$$P(B_S, D) = P(B_S) \cdot \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i \Leftrightarrow 1)!}{(N_{ij} + r_i \Leftrightarrow 1)!} \cdot \prod_{k=1}^{r_i} N_{ijk}!. \qquad \textbf{4.1}$$

A derivation of Formula 4.1 can be found in [24]. Note that if $N_{ij} = 0$ then also $N_{ijk} = 0$ for $k = 1, \ldots, r_i$, and $\frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \cdot \prod_{k=1}^{r_i} N_{ijk}! = 1$.

In the right-hand side of Formula 4.1, the term $P(B_S)$ denotes the prior probability of the network structure B_S. In this term, information about the 'real' network structure prior to observation of the database can be incorporated. For example, if a domain expert suggests existence of a specific arc or a specific direction of an arc, then network structures that adhere to this suggestion are given a higher prior

[1] We use $P(B_{S_1}|D)$ as a short notation for $Pr($ the network structure $= B_{S_1}|$ the database $= D)$.

probability. If no prior information is available, $P(B_S)$ is chosen to be a uniform distribution. The other terms in the right-hand side of Formula 4.1 represent how well the network structure B_S fits the database based on the number of appearances of cases; however, these terms are not very intuitive.

In applications of the Bayesian measure proposed by Cooper and Herskovits, generally the logarithm of Formula 4.1 is used for pragmatical reasons: even for small databases with N cases, numbers like $N!$ tend to give computational problems (note that $100! \approx 10^{160}$).

4.3 **Definition** Let V be a set of variables, and let $\mathcal{D}$ be the set of all possible databases over V. Let $\mathcal{B}_S$ be the set of all possible network structures over V. Then, the *Bayesian measure* is the function $\mathcal{B}_S \times \mathcal{D} \to \mathbb{R}$ defined by

$$B(B_S, D) = \log P(B_S, D)$$

for all $B_S \in \mathcal{B}_S$ and $D \in \mathcal{D}$, where $P(B_S, D)$ as in Formula 4.1. □

Note that the Bayesian measure is a quality measure, since for each network structure B_S and each database D the probability $P(B_S, D)$ is a value in the unit interval and the logarithm of this value is not positive.

Related Bayesian measures have been proposed [14, 52, 53, 102]. Basically, in these approaches it is assumed that the distribution over the assessment functions is a Dirichlet distribution. Since they seem to have a marginal influence on the quality measure derived and they rely too much on statistics, we will restrict ourselves in this thesis to the assumption of Cooper and Herskovits.

4.2.2 An Information Criterion Approach

As opposed to the Bayesian approach to model selection, the information criterion approach stems from a frequentist point of departure. The basic idea is to take the network structure with the best fit to the database penalized by the number of values that have to be specified to define assessment functions for the network structure.

The fit of a network structure B_S and a database D is taken to be proportional to the probability that the database was generated by the distribution represented by the Bayesian belief network $B = (B_S, B_P)$; the assessment functions in B_P are taken to be frequentist estimates calculated from the database, that is, for every variable v_i we have

4.2
$$\gamma_{v_i}(v_i = x_{ik} | \pi_i = x_{\pi_i j}) = \frac{N_{ijk}}{N_{ij}}$$

for all values x_{ik} and configurations $x_{\pi_i j}$. The probability of the database D given the Bayesian belief network B therefore is

$$P(D|B) = \prod_{x_V \in D} Pr(V = x_V),$$

where Pr is the joint probability distribution represented by B. This probability equals

$$P(D|B) = \prod_{x_V \in D} \prod_{i=1}^{n} \gamma_{v_i}(v_i = x_{v_i} | \pi_i = x_{\pi_i}).$$

By reordering terms, we can write this probability as

$$P(D|B) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \gamma_{v_i}(v_i = x_{ik} | \pi_i = x_{\pi_i j})^{N_{ijk}}.$$

Substitution of the probability estimates from Formula 4.2 gives

$$P(D|B) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \left(\frac{N_{ijk}}{N_{ij}} \right)^{N_{ijk}},$$

where by convention $\frac{0}{0}^0 = 1$. Taking the logarithm gives

$$\log P(D|B) = N \cdot \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}}, \tag{4.3}$$

where by convention $0 \log 0 = 0$. The result equals $\Leftrightarrow N \cdot H(B_S, D)$ where $H(B_S, D)$ is the *entropy* of B_S and D. Entropy is a non-negative measure of uncertainty. Entropy is maximal when uncertainty is maximal and zero when there is complete knowledge as to its value [67]. Intuitively, the more information is given by the database, that is, the closer the probabilities estimated from the database are to 1, the lower the entropy. Note that in general a network structure B_S with large parent sets will have a lower entropy than network structures that are subgraphs of B_S with small parent sets since a probability distribution can be more accurately described with Bayesian belief networks containing these network structures; the larger a parent set, the less cases in the database there are per estimated probability, and the more often a probability is estimated as 1.

The number of probabilities that have to be estimated from the database to define the assessment function for a single variable increases exponentially with the cardinality of its parent set. With every estimate, an error is introduced in the assessment function. This error will be larger when the amount of data the estimate is based on is smaller, and the more estimates have to be made the smaller the average number of cases in the database these estimates can be based on. To model this effect, a penalty term is added to the entropy term. This penalty term depends on the number of probabilities that have to be estimated to define all assessment functions for the network structure under consideration. Let B_S be a network structure over V. To define an assessment function for variable v_i with r_i values and no parents in the network structure, only $r_i \Leftrightarrow 1$ values have to be specified. Because the assessment function of v_i has to sum to unity for a given configuration of the parent set, the r_ith value follows directly. If the parent set is not empty, it suffices to specify $r_i \Leftrightarrow 1$ values q_i times to define the assessment function for variable v_i. So, the number of values to be specified for B is

$$K = \sum_{i=1}^{n} (r_i \Leftrightarrow 1) \cdot q_i.$$

We will refer to K as the *number of parameters* of B. We also write that K is the number of parameters of B_S to denote the number of parameters of B. Since the entropy term $\Leftrightarrow N \cdot H(B_S, D)$ increases with increasing database sizes, the penalty term may be compensated. In the penalty term, K is multiplied by a non-negative penalty function f that is a function of the database size N. For $f(N) > 0$, the penalty term in general is larger when the network structures contain larger parent sets.

To incorporate prior information about network structures, a term $P(B_S)$ may be added to the entropy and the penalty term to obtain the information criterion.

4.4 **Definition** Let V be a set of variables, and let $\mathcal{D}$ be the set of all possible databases over V. Let $\mathcal{B}_S$ be the set of all possible network structures over V. Let r_i, q_i, N_{ijk}, and N_{ij} be as before. Then, the *information criterion* with *penalty function* f is a function $\mathcal{B}_S \times \mathcal{D} \to \mathbb{R}$ defined by

4.4 $$I_f(B_S, D) = \log P(B_S) \Leftrightarrow N \cdot H(B_S, D) \Leftrightarrow K \cdot f(N),$$

where $H(B_S, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \Leftrightarrow \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}}$, $K = \sum_{i=1}^{n} (r_i \Leftrightarrow 1) \cdot q_i$, and f is a non-negative real-valued function. □

Many different penalty functions have been proposed in the literature. With $f(N) = 0$, we get the *maximum likelihood* information criterion. The information criterion with penalty function $f(N) = 1$ is known as the *Akaike information criterion*, or simply *AIC* [3]. The information criterion with penalty function $f(N) = \frac{1}{2} \log N$ is known as the *BIC* or *Schwarz criterion* [94] and coincides with the measure based on the minimum description length principle that we will review in the next section. Hannan and Quinn [51] suggest $f(N) = c \log \log N$ for some positive constant c. Basically, to get a consistent information criterion, it has been suggested that for N approximating infinity, $f(N)$ should approximate infinity, and $f(N)/N$ should approximate zero [12, 75]. Based on empirical evidence, however, practical values of $f(N)$ between 2 and 6 have proven to perform well in model-selecting tasks [35].

In the prior probability of the network structure information about the 'real' network structure can be incorporated again, in the same way as for the Bayesian measure. Since frequentist statisticians usually do not model such information, this term is often omitted in literature [3, 21, 94]. We have decided to add this term to the information criterion because it allows for modeling prior information on the one hand and makes comparison with the Bayesian measure more convenient on the other hand.

The information criterion consists of three terms:

- the prior probability of the network structure;
- the entropy of the network structure and the database; and
- a penalty.

Figure 4.1 shows the interaction between these terms in the situation that no prior information is available, that is, where $P(B_S)$ is constant for all network structures B_S. The x-axis shows the number of parameters K and the y-axis the various components of an information criterion. For small values of K, the value of the penalty term is small and the entropy term $\Leftrightarrow N \cdot H(B_S, D)$ is large. With increasing K, the entropy term usually increases and the penalty term linearly decreases. Note

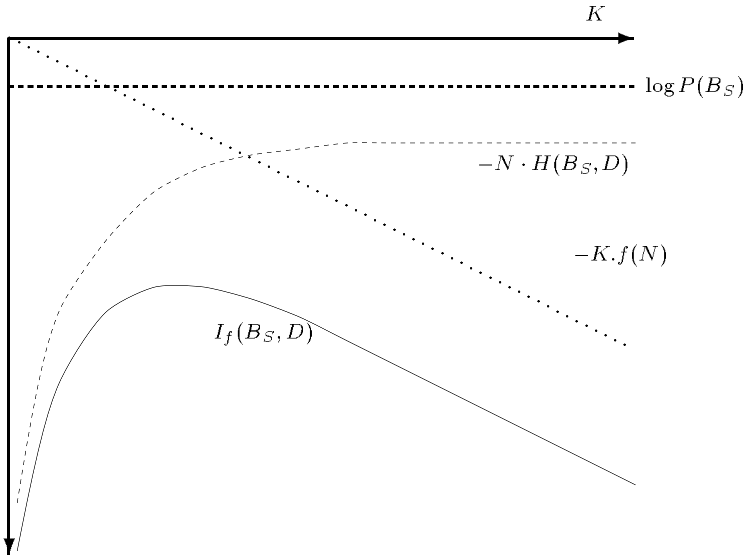


Figure 4.1: Influence of various components in an information criterion varying with the number of parameters K.

that the slope of the line $\Leftrightarrow K \cdot f(N)$ depends on the specific choice of f and is arbitrarily chosen in the figure. As long as the entropy term increases more than the penalty term decreases, the information criterion will increase up to a point where the increase in entropy is dominated by the penalty term.

Note that the information criterion is a quality measure, because it assigns a non-positive value to each network structure and a database. This observation follows from the fact that all three terms in Formula 4.4 are non-positive. Other selection criteria exist that resemble Formula 4.4 but have, for example, another term added to it [61]. These criteria are out of the scope of this thesis.

4.2.3 A Minimum Description Length Approach

Another approach to measure the quality of a network structure and a database is the minimum description length principle [89, 90]. This principle stems from coding theory where it is used for encoding a string of symbols with as few bits as possible. The basic idea is to compress the string using a probability distribution over all possible strings; the most likely strings are encoded by short messages, and the least likely are encoded by long messages resulting in an as short as possible average message length. The encoding then consists of two parts, the description of the probability distribution Pr used for compression, and the compressed string. The description length of the string of symbols is the sum of the lengths of the distribution and the compressed string. The minimum description length principle selects for a given string of symbols the probability distribution for which the description length is minimal. To apply this principle to network structure learning from data, we consider encoding a database D by compressing it using the distribution represented by a Bayesian belief network. We then simply select the network structure B_S for which the description length of D using the distribution represented by the be added to the entropy and the penalty term to obtain the complete information criterion.

The description length of a database D using the probability distribution defined by the Bayesian belief network B is now given by in the following definition.

4.5 **Definition** Let V be a set of variables, and let $\mathcal{D}$ be the set of all possible databases over V. Let $\mathcal{B}_{\mathcal{S}}$ be the set of all possible network structures over V. Let r_i, q_i, N_{ijk}, and N_{ij} be as before. Then, the *description length* is the function $\mathcal{B}_{\mathcal{S}} \times \mathcal{D} \rightarrow \mathbb{R}$ defined by

4.5
$$L(B_S, D) = \log P(B_S) \Leftrightarrow N \cdot H(B_S, D) \Leftrightarrow \frac{1}{2} K \cdot \log N,$$

where $H(B_S, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \Leftrightarrow \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}}$ and $K = \sum_{i=1}^{n} (r_i \Leftrightarrow 1) \cdot q_i$. □

Though it does not stem from the minimum description length principle, the first term $\log P(B_S)$ is included to capture prior knowledge and to make comparison with the Bayesian measure more convenient. Note that the length of the description of the network structure B_S is omitted from the description length, because it is a constant for a given set of variables and therefore for the database and we are mainly interested in the difference of description length of various structures. Also note that the description lengths are preceded by a minus sign because we want to maximize the minimum description length with the objective of maximizing quality of network structures and databases.

Like information criteria, the description length consists of three terms:

- the prior probability of the network structure;
- the description length of the database encoded; and
- the description length of the encoding distribution.

The three terms of the description length interact with each other as depicted in Figure 4.1. Assume that no prior information is available, that is, assume that the prior on the network structures $Pr(B_S)$ is constant. The fewer arcs are comprised in the network structure B_S, the shorter the description of the assessment functions. However, in a Bayesian belief network containing a network structure with only a few arcs not much detailed information can be represented in the assessment functions. Therefore, the fewer arcs are comprised in B_S, the longer the description of the database. On the other hand, the more arcs there are in the network structure, the longer the description of the assessment functions and the shorter the description of the database.

Note that the description length is a quality measure because it assigns a non-positive value to each network structure and database. This observation follows from the fact that all three terms in Formula 4.5 are non-positive. In the sequel, we will refer to the description length measure as the *MDL measure*.

4.2.4 Comparing Quality Measures

In this section, we compare the quality measures proposed in the previous sections with each other, to get an insight in the resemblance and differences in the behavior of these quality measures. If measures share the same behavior, then it can be justified that they can replace each other. For example, for hypothesis testing there is a Bayesian justification to apply the Bayesian measure. If it turns out that other quality measures show the same behavior, then it can be justified to use these quality measures for hypotheses testing as well.

Though the philosophy of the minimum description length principle is completely different from that of the information criterion, the two measures bear a close resemblance. In fact, the description length is equal to the information criterion with penalty function $f(N) = \frac{1}{2}\log(N)$.

Because of the similarity of the information criterion and the MDL measure, we will focus on the relation between the Bayesian measure and the MDL measure. The following theorem tells that for any network structure and some databases these two measures yield approximately equal values.

4.1 **Theorem** Let V be a set of variables. Let B_S be a network structure over V, and let D be a database over V such that each configuration of every parent set of B_S occurs in the database. Let $B(B_S, D)$ be the Bayesian measure of B_S and D, and let $L(B_S, D)$ be the MDL measure of B_S and D. Then,

$$L(B_S, D) = B(B_S, D) + O(1) \tag{4.6}$$

where the $O(1)$ is with respect to N. □

Proof: Let r_i, q_i, N_{ijk}, and N_{ij} be as before. By the definition of $B(B_S, D)$, we have

$$B(B_S, D) = \log P(B_S) + \sum_{i=1}^{n}\sum_{j=1}^{\eta_i}\left\{\log(r_i \Leftrightarrow 1)! \Leftrightarrow \log(N_{ij} + r_i \Leftrightarrow 1)! + \sum_{k=1}^{r_i}\log N_{ijk}!\right\}, \tag{4.7}$$

where η_i is the number of unique configurations of the parent set π_i that occur in the database. From the condition stated in the theorem we have that η_i equals q_i. However, to show where the Bayesian measure and the MDL measure differ if the condition does not hold, we write η_i for the moment. Now, consider the contribution of one variable v_i and one configuration of its parent set π_i to the double summation in the right-hand side of the above equality which equals

$$\log(r_i \Leftrightarrow 1)! \Leftrightarrow \log(N_{ij} + r_i \Leftrightarrow 1)! + \sum_{k=1}^{r_i}\log N_{ijk}!.$$

This contribution can be written as,

$$\log(r_i \Leftrightarrow 1)! \Leftrightarrow \log\left((N_{ij} + 1)\cdot \ldots \cdot (N_{ij} + r_i \Leftrightarrow 1)\right) \Leftrightarrow \log N_{ij}! + \sum_{k=1}^{r_i}\log N_{ijk}!. \tag{4.8}$$

We now apply Stirling's formula $x! \approx \sqrt{2\pi x}(\frac{x}{e})^x$ to the last two terms of Expression

4.8, giving

$$\Leftrightarrow \log \sqrt{2\pi N_{ij}} \left(\frac{N_{ij}}{e}\right)^{N_{ij}} + \sum_{k=1}^{r_i} \log \sqrt{2\pi N_{ijk}} \left(\frac{N_{ijk}}{e}\right)^{N_{ijk}} . \tag{4.9}$$

Note that, since the approximation $\sqrt{2\pi x}(\frac{x}{e})^x$ has a relative error of about $\frac{1}{12x}$ [49], we introduce an $O(1)$ error with respect to N. Expression 4.9 equals

$$\Leftrightarrow \frac{1}{2} \log 2\pi \Leftrightarrow \left(N_{ij} + \frac{1}{2}\right) \log N_{ij} + N_{ij} \log e +$$
$$\sum_{k=1}^{r_i} \left\{ \frac{1}{2} \log 2\pi + \left(N_{ijk} + \frac{1}{2}\right) \log N_{ijk} \Leftrightarrow N_{ijk} \log e \right\} .$$

By exploiting that $\sum_{k=1}^{r_i} N_{ijk} = N_{ij}$ by definition and subsequently grouping terms, the $\log e$ terms cancel out, and we find

$$\sum_{k=1}^{r_i} \frac{1}{2} \log N_{ijk} \Leftrightarrow \frac{1}{2} \log N_{ij} + \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} + \frac{r_i \Leftrightarrow 1}{2} \log 2\pi .$$

For a large enough database, that is, for N large enough, the last term of this expression is negligible; we therefore omit this term, once more introducing an $O(1)$ error. Substitution of the result for the last two terms of Expression 4.8 gives

$$\log(r_i \Leftrightarrow 1)! \Leftrightarrow \log(N_{ij}+1) \cdot \ldots \cdot (N_{ij}+r_i \Leftrightarrow 1) + \sum_{k=1}^{r_i} \frac{1}{2} \log N_{ijk} \Leftrightarrow \frac{1}{2} \log N_{ij} + \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} . \tag{4.10}$$

The $\log(r_i \Leftrightarrow 1)!$ term is also negligible for a large enough database; we delete this term, again introducing an $O(1)$ error.

Now consider the term $\Leftrightarrow \log(N_{ij}+1) \ldots (N_{ij}+r_i \Leftrightarrow 1)$ of Expression 4.10. This term can be approximated by $\Leftrightarrow \log N_{ij}^{r_i - 1}$. By this approximation, an error $\sum_{p=1}^{r_i-1} \log \frac{N_{ij}+p}{N_{ij}}$ is introduced. Using $\log \frac{N_{ij}+p}{N_{ij}} < \log p \leq \log(r_i \Leftrightarrow 1)$, for $1 \leq p \leq r_i \Leftrightarrow 1$ we find for this error that $\sum_{p=1}^{r_i-1} \log \frac{N_{ij}+p}{N_{ij}} < (r_i \Leftrightarrow 1) \cdot \log(r_i \Leftrightarrow 1)$. Since $(r_i \Leftrightarrow 1) \cdot \log(r_i \Leftrightarrow 1)$ is a constant with respect to N, the approximation of $\Leftrightarrow \log(N_{ij} + 1) \ldots (N_{ij} + r_i \Leftrightarrow 1)$ by $\Leftrightarrow \log N_{ij}^{r_i-1}$ introduces yet another $O(1)$ error.

Expression 4.10, therefore, can be approximated by

$$\begin{aligned}
&\Leftrightarrow \log N_{ij}^{r_i - 1} + \sum_{k=1}^{r_i} \frac{1}{2} \log N_{ijk} \Leftrightarrow \frac{1}{2} \log N_{ij} + \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} \\
= &\quad \log \frac{\sqrt{\prod_{k=1}^{r_i} N_{ijk}}}{N_{ij}^{r_i - 1} \sqrt{N_{ij}}} + \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} \\
= &\quad \log \left(\frac{\sqrt{\prod_{k=1}^{r_i} \frac{N_{ijk}}{N}}}{\left(\frac{N_{ij}}{N}\right)^{r_i - \frac{1}{2}}} \cdot \frac{N^{\frac{1}{2} r_i}}{N^{r_i - \frac{1}{2}}} \right) + \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} \\
= &\quad \sum_{k=1}^{r_i} \frac{1}{2} \log \frac{N_{ijk}}{N} \Leftrightarrow \left(r_i \Leftrightarrow \frac{1}{2}\right) \log \frac{N_{ij}}{N} + \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} \Leftrightarrow \frac{r_i \Leftrightarrow 1}{2} \log N \\
\approx &\quad \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} \Leftrightarrow \frac{r_i \Leftrightarrow 1}{2} \log N
\end{aligned}$$

The last approximation is allowed, since by the strong law of large numbers $\frac{N_{ijk}}{N}$ approximates a probability $Pr(v_i = x_{ik}, \pi_i = x_{\pi_i j})$ and $\frac{N_{ij}}{N}$ approximates $Pr(\pi_i = x_{\pi_i j})$. Note that such probability exists by the basic assumption that all cases in a database are independent. These terms are negligible for N large enough. By this approximation, again an $O(1)$ error is introduced.

Recall that, so far, we performed the approximation for a specific value of i and j. Summation over j of the above expression gives,

$$N \cdot \sum_{j=1}^{\eta_i} \sum_{k=1}^{r_i} \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}} \Leftrightarrow \frac{\eta_i(r_i \Leftrightarrow 1)}{2} \log N.$$

Further summation over i gives

$$N \sum_{i=1}^{n} \sum_{j=1}^{\eta_i} \sum_{k=1}^{r_i} \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}} \Leftrightarrow \sum_{i=1}^{n} \frac{\eta_i(r_i \Leftrightarrow 1)}{2} \log N. \tag{4.11}$$

Now recall from the conditions of our theorem that all possible configurations of the parent sets of B_S occur in the database and therefore, $\eta_i = q_i$. Expression 4.11 therefore equals

$$\Leftrightarrow N \cdot H(B_S, D) \Leftrightarrow \frac{1}{2} K \cdot \log N,$$

from which the desired result is obtained. We would like to stress once more that every approximation made in the course of the derivation has introduced an error of $O(1)$ with respect to N. □

From this theorem and the definition of the MDL measure, we have that the Bayesian measure and MDL measure will yield approximately the same results for databases where all configurations of parent sets occur at least once. So, if the Bayesian measure prefers a network structure B_S over $B_{S'}$, then the MDL measure will prefer B_S over $B_{S'}$ most of the time.

Note, however, that the two measures will not always prefer the same network structures due to the constant $C_{B_{S,D}}$, which may differ among network structures and therefore makes the measures slightly different. Also, for databases in which not all possible configurations of the parent sets of a network structure occur, different results will be obtained. Further, from the approximation of $\log N_{ijk}$ and $\log N_{ij}$ by $\log N$, it is easily seen that the MDL measure assigns a larger weight to the cost of estimating parameters (the $\frac{1}{2} K \log N$-term) than the Bayesian measure. When not all configurations of all parent sets occur in the database, still all these configurations are accounted for by the MDL measure but not by the Bayesian measure. As a result, the MDL measure may prefer a network with fewer arcs than the Bayesian measure prefers.

4.2.5 Score equivalence

One need not be interested in the represented distribution but instead one may be interested in the causal structure underlying the domain. Now, in a network structure only those arcs that have the same direction in any equivalent network

a	b
0	0
0	0
0	0
0	1
1	0
1	0
1	1
1	1

Table 4.1: A database of cases over two binary variables.

structure represent causal influences [104]. From this point of view, a network structure is as valuable as any of its equivalent network structures. So, it is desirable that quality measures assign the same quality to equivalent network structures when no prior information is available (see also [53]). We formalize this property with the term score equivalence.

4.6 **Definition** Let V be a set of variables, and let $\mathcal{D}$ be the set of all possible databases over V. Let $\mathcal{B}_S$ be the set of all possible network structures over V. Let Q be a quality measure. Then Q is *score equivalent* if for all $D \in \mathcal{D}$ and $B_S, B'_S \in \mathcal{B}_S$, we have that $Q(B_S, D) = Q(B'_S, D)$ if the network structures B_S and B'_S are equivalent. □

Unfortunately, the Bayesian measure is not score equivalent if a uniform prior distribution over the network structures is assumed. Consider the database in Table 4.1 over two binary variables a and b, and the network structures B_{S_1} being $a \to b$ and B_{S_2} being $b \to a$. It will be evident that both structures represent the same set of independencies. Yet,

$$P(B_{S_1}, D) = P(B_{S_1}) \frac{(2 \Leftrightarrow 1)!}{(8 + 2 \Leftrightarrow 1)!} 4!4! \frac{(2 \Leftrightarrow 1)!}{(4 + 2 \Leftrightarrow 1)!} \frac{(2 \Leftrightarrow 1)!}{(4 + 2 \Leftrightarrow 1)!} 3!1!2!2! = P(B_{S_1}) \frac{24}{25} \frac{1}{9!}$$

and

$$P(B_{S_2}, D) = P(B_{S_2}) \frac{(2 \Leftrightarrow 1)!}{(8 + 2 \Leftrightarrow 1)!} 5!3! \frac{(2 \Leftrightarrow 1)!}{(5 + 2 \Leftrightarrow 1)!} \frac{(2 \Leftrightarrow 1)!}{(3 + 2 \Leftrightarrow 1)!} 3!1!2!2! = P(B_{S_2}) \frac{1}{9!}.$$

So, if we assume both structures equiprobable we have that $P(B_{S_1}, D) = \frac{24}{25} P(B_{S_2}, D)$. For measures based on other Bayesian approaches [14, 52, 102], score equivalence does hold. In these approaches a non-uniform prior distribution over the assessment functions is used and formulas differing slightly from 4.1 are obtained.

In the remainder of this section, we will show that the information criteria, and specifically the MDL measure, assign equal quality to network structures that represent the same independency model. Before doing so, we state some useful properties of information criteria.

For a network structure, if a single arc-reversal operation is applied on a pair of nodes u and v that are adjacent, then the quality of the structure does not change

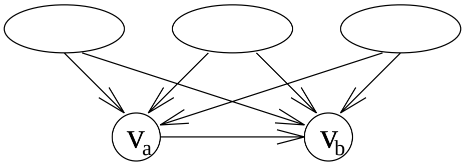


Figure 4.2: A network structure with $v_a \in \pi_b$ and $\pi_a = \pi_b \backslash v_a$.

if the parent set of u equals the parent set of v (excluding u itself). First, we examine the influence of such an arc reversal on the entropy; after this, we examine its influence on the number of parameters.

4.1 **Lemma** Let V be a set of variables, and let D be a database over V. Let B_S be a network structure over V. Furthermore, let v_a and v_b be two nodes in B_S such that either $v_a \notin \pi_b$ or $v_a \in \pi_b$ and $\pi_a = \pi_b \backslash v_a$ and let $B_{S'} = arcr(B_S, v_a, v_b)$. Then,

$$H(B_S, D) = H(B_{S'}, D),$$

where H is the entropy as defined in Formula 4.3. □

Proof: We distinguish between the two cases. In the case where $v_a \notin \pi_b$, we have by definition that $B_S = B_{S'}$ and the lemma is trivially true. In the remainder of the proof, we address the case where $v_a \in \pi_b$ and $\pi_a = \pi_b \backslash v_a$; Figure 4.2 sketches the basic idea.

Let r_i, q_i, N_{ijk}, and N_{ij} be as before for D and B_S, and let r'_i, q'_i, N'_{ijk}, and N'_{ij} be likewise for D and $B_{S'}$. Note that $r_i = r'_i$ for all $i = 1, \ldots, n$. For notational convenience, we now prove that $N \cdot H(B_S, D) = N \cdot H(B_{S'}, D)$. By definition, we have

$$N \cdot H(B_S, D) = \Leftrightarrow \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}.$$

Since $q'_i = q_i$, $\pi_i = \pi'_i$, $N'_{ij} = N_{ij}$ and $N'_{ijk} = N_{ijk}$ for all nodes $v_i \notin \{v_a, v_b\}$ we have that the entropy term equals

4.12 $$\Leftrightarrow \sum_{i=1, i \neq a, i \neq b}^{n} \sum_{j=1}^{q'_i} \sum_{k=1}^{r_i} N'_{ijk} \log \frac{N'_{ijk}}{N'_{ij}} \Leftrightarrow \sum_{j=1}^{q_a} \sum_{k=1}^{r_a} N_{ajk} \log \frac{N_{ajk}}{N_{aj}} \Leftrightarrow \sum_{j=1}^{q_b} \sum_{k=1}^{r_b} N_{bjk} \log \frac{N_{bjk}}{N_{bj}}.$$

Consider the last two terms of Expression 4.12. These terms equal

4.13 $$\sum_{j=1}^{q_a} \sum_{k=1}^{r_a} N_{ajk} \log N_{aj} \Leftrightarrow \sum_{j=1}^{q_a} \sum_{k=1}^{r_a} N_{ajk} \log N_{ajk} + \sum_{j=1}^{q_b} \sum_{k=1}^{r_b} N_{bjk} \log N_{bj} \Leftrightarrow \sum_{j=1}^{q_b} \sum_{k=1}^{r_b} N_{bjk} \log N_{bjk}.$$

Now consider the third term of Expression 4.13. Using $N_{bj} = \sum_{k=1}^{r_b} N_{bjk}$, we find that $\sum_{j=1}^{q_b} \sum_{k=1}^{r_b} N_{bjk} \log N_{bj} = \sum_{j=1}^{q_b} N_{bj} \log N_{bj}$. From $\pi_b = \pi_a v_a$, we have that for every N_{bj} there is a unique index j' together with an index k such that $N_{aj'k} = N_{bj}$. So, $\sum_{j=1}^{q_b} N_{bj} \log N_{bj}$ can be written as $\sum_{j=1}^{q_a} \sum_{k=1}^{r_a} N_{ajk} \log N_{ajk}$. Substitution in Expression 4.13 gives

$$\sum_{j=1}^{q_a} \sum_{k=1}^{r_a} N_{ajk} \log N_{aj} \Leftrightarrow \sum_{j=1}^{q_a} \sum_{k=1}^{r_a} N_{ajk} \log N_{ajk} + \sum_{j=1}^{q_a} \sum_{k=1}^{r_a} N_{ajk} \log N_{ajk} \Leftrightarrow \sum_{j=1}^{q_b} \sum_{k=1}^{r_b} N_{bjk} \log N_{bjk}.$$

The middle two terms cancel out, yielding

$$\sum_{j=1}^{q_a}\sum_{k=1}^{r_a} N_{ajk}\log N_{aj} \Leftrightarrow \sum_{j=1}^{q_b}\sum_{k=1}^{r_b} N_{bjk}\log N_{bjk}. \tag{4.14}$$

For the first term of Expression 4.14 we once more have $\sum_{j=1}^{q_a}\sum_{k=1}^{r_a} N_{ajk}\log N_{aj}$ equals $\sum_{j=1}^{q_a} N_{aj}\log N_{aj}$. From $\pi'_b = \pi_a$, we further have that $q'_b = q_a$ and $N'_{bj} = N_{aj}$; so, we can write the sum $\sum_{j=1}^{q_a} N_{aj}\log N_{aj}$ as $\sum_{j=1}^{q'_b} N'_{bj}\log N'_{bj}$, which is equal to $\sum_{j=1}^{q'_b}\sum_{k=1}^{r'_b} N'_{bjk}\log N'_{bj}$.

Now, we consider the second term $\sum_{j=1}^{q_b}\sum_{k=1}^{r_b} N_{bjk}\log N_{bjk}$ of Expression 4.14. Recall that N_{bjk} is the number of cases in which the parent set π_b of node v_b takes the configuration $x_{\pi_b j}$ and node v_b takes the value x_{bk}; likewise, $N'_{aj'k'}$ is the number of cases in which π'_a takes the configuration $x'_{\pi_a j'}$ and v_a takes the value $x'_{bk'}$. Since $\pi_b v_b = \pi'_a v_a$, we have that there are indices j' and k' such that $N_{bjk} = N'_{aj'k'}$. So, $\sum_{j=1}^{q_b}\sum_{k=1}^{r_b} N_{bjk}\log N_{bjk}$ can be written as $\sum_{j=1}^{q'_a}\sum_{k=1}^{r'_a} N'_{ajk}\log N'_{ajk}$. Expression 4.14 therefore equals

$$\sum_{j=1}^{q'_b}\sum_{k=1}^{r'_b} N'_{bjk}\log N'_{bj} \Leftrightarrow \sum_{j=1}^{q'_a}\sum_{k=1}^{r'_a} N'_{ajk}\log N'_{ajk}. \tag{4.15}$$

Now observe that

$$\sum_{j=1}^{q'_a}\sum_{k=1}^{r'_a} N'_{ajk}\log N'_{aj} \Leftrightarrow \sum_{j=1}^{q'_b}\sum_{k=1}^{r'_b} N'_{bjk}\log N'_{bjk} = 0,$$

using $\pi'_a = \pi'_b v_b$. Adding this term to Expression 4.15 gives

$$\sum_{j=1}^{q'_b}\sum_{k=1}^{r'_b} N'_{bjk}\log N'_{bj} \Leftrightarrow \sum_{j=1}^{q'_b}\sum_{k=1}^{r'_b} N'_{bjk}\log N'_{bjk} + \sum_{j=1}^{q'_a}\sum_{k=1}^{r'_a} N'_{ajk}\log N'_{aj} \Leftrightarrow \sum_{j=1}^{q'_a}\sum_{k=1}^{r'_a} N'_{ajk}\log N'_{ajk}.$$

Using $\log x \Leftrightarrow \log y = \log\frac{x}{y}$ we get,

$$\Leftrightarrow \sum_{j=1}^{q'_b}\sum_{k=1}^{r'_b} N'_{bjk}\log\frac{N'_{bjk}}{N'_{bj}} \Leftrightarrow \sum_{j=1}^{q'_a}\sum_{k=1}^{r'_a} N'_{ajk}\log\frac{N'_{ajk}}{N'_{aj}}.$$

Substituting this result in Expression 4.12 gives,

$$\Leftrightarrow \sum_{i=1, i\neq a, j\neq b}^{n}\;\sum_{j=1}^{q'_i}\sum_{k=1}^{r'_i} N'_{ijk}\log\frac{N'_{ijk}}{N'_{ij}} \Leftrightarrow \sum_{j=1}^{q'_a}\sum_{k=1}^{r'_a} N'_{ajk}\log\frac{N'_{ajk}}{N'_{aj}} \Leftrightarrow \sum_{j=1}^{q'_b}\sum_{k=1}^{r'_b} N'_{bjk}\log\frac{N'_{bjk}}{N'_{bj}},$$

and by reordering terms, this is,

$$\Leftrightarrow \sum_{i=1}^{n}\sum_{j=1}^{q'_i}\sum_{k=1}^{r'_i} N'_{ijk}\log\frac{N'_{ijk}}{N'_{ij}},$$

which by definition is $N \cdot H(B_{S'}, D)$. □

Now observe that the condition in Lemma 4.1, that is, that either $v_a \notin \pi_b$ or $v_a \in \pi_b$ and $\pi_a = \pi_b \backslash v_a$, is the same as the condition that B_S and $B_{S'}$ are equivalent. So the lemma states that a single arc reversal on a network structure B_S has no influence on the entropy of the structure and a database, as long as the obtained network structure $B_{S'}$ is equivalent with B_S. The following lemma gives a similar result for the number of parameters that have to be assessed for the two network structures B_S and $B_{S'}$.

4.2 **Lemma** Let V be a set of variables, and let D be a database over V. Let B_S be a network structure over V. Furthermore, let v_a and v_b be two nodes in B_S such that either $v_a \notin \pi_b$ or $v_a \in \pi_b$ and $\pi_a = \pi_b \backslash v_a$ and let $B_{S'} = arcr(B_S, v_a, v_b)$. Then,

$$K = K',$$

where K and K' are the number of parameters of B_S and $B_{S'}$, respectively. □

Proof: We distinguish between the two cases. In the case where $v_a \notin \pi_b$ we have by definition that $B_S = B_{S'}$ and the lemma is trivially true. In the remainder of the proof, we address the case where $v_a \in \pi_b$ and $\pi_a = \pi_b \backslash v_a$.

Let r_i, q_i, N_{ijk}, and N_{ij} be as before for D and B_S, and let r'_i, q'_i, N'_{ijk}, and N'_{ij} be likewise for D and $B_{S'}$. By definition of K and K', we have

$$K \Leftrightarrow K' = \sum_{i=1}^{n} \{q_i(r_i \Leftrightarrow 1) \Leftrightarrow q'_i(r'_i \Leftrightarrow 1)\}.$$

We recall that $r'_i = r_i$ for all $i = 1, \ldots, n$. In addition, we have for all $i = 1, \ldots, n$, $i \neq a$, $i \neq b$, that $q'_i = q_i$. From these observations, we have

$$K \Leftrightarrow K' = q_a(r_a \Leftrightarrow 1) + q_b(r_b \Leftrightarrow 1) \Leftrightarrow q'_a(r_a \Leftrightarrow 1) \Leftrightarrow q'_b(r_b \Leftrightarrow 1).$$

By definition, we have, $q_a = \prod_{v_j \in \pi_a} r_j$ and $q_b = \prod_{v_j \in \pi_b} r_j$; a similar observation holds for q'_a and q'_b. Substitution gives

$$K \Leftrightarrow K' = \prod_{v_j \in \pi_a} r_j(r_a \Leftrightarrow 1) + \prod_{v_j \in \pi_b} r_j(r_b \Leftrightarrow 1) \Leftrightarrow \prod_{v_j \in \pi'_a} r_j(r_a \Leftrightarrow 1) \Leftrightarrow \prod_{v_j \in \pi'_b} r_j(r_b \Leftrightarrow 1)$$

$$= (\prod_{v_j \in \pi_a} r_j \Leftrightarrow \prod_{v_j \in \pi'_a} r_j)(r_a \Leftrightarrow 1) + (\prod_{v_j \in \pi_b} r_j \Leftrightarrow \prod_{v_j \in \pi'_b} r_j)(r_b \Leftrightarrow 1).$$

Since $\pi'_a = \pi_a b$ and $\pi'_b = \pi_b \backslash a = \pi_a$, we find

$$K \Leftrightarrow K' = \prod_{v_j \in \pi_a} r_j(1 \Leftrightarrow r_b)(r_a \Leftrightarrow 1) + \prod_{v_j \in \pi_a} r_j(r_a \Leftrightarrow 1)(r_b \Leftrightarrow 1) = 0.$$

We conclude that $K = K'$. □

The previous two lemmas together indicate that a single arc reversal does not change the quality of a network structure if an information criterion is used.

4.3 **Lemma** Let V be a set of variables, and let D be a database over V. Let B_S be a network structure over V. Let the prior probability distribution on network structures over V be uniform. Now, let v_a and v_b be two nodes in B_S such that either $v_a \notin \pi_b$ or $v_a \in \pi_b$ and $\pi_a = \pi_b \backslash v_a$, and let $B_{S'} = arcr(B_S, v_a, v_b)$. Then

$$I_f(B_S, D) = I_f(B_{S'}, D),$$

where $I_f(B_S, D)$ is an information criterion with penalty function f as defined in Definition 4.4. □

Proof: We have to show that $\log P(B_S) \Leftrightarrow N \cdot H(B_S, D) \Leftrightarrow K \cdot f(N) = \log P(B_{S'}) \Leftrightarrow N \cdot H(B_{S'}, D) \Leftrightarrow K' \cdot f(N)$, where N, K and K' as before. We will prove the equality by showing that $\log P(B_S) = \log P(B_{S'})$, $N \cdot H(B_S, D) = N \cdot H(B_{S'}, D)$, and $K \cdot f(N) = K' \cdot f(N)$. From the uniform prior distribution over all network structures, we have that $P(B_S)$ equals $P(B_{S'})$, and therefore that $\log P(B_S)$ equals $\log P(B_{S'})$. Using Lemma 4.1, we find that $H(B_S, D) = H(B_{S'}, D)$, so $N \cdot H(B_S, D)$ equals $N \cdot H(B_{S'}, D)$. Using Lemma 4.2, we find that $K = K'$, and therefore that $K \cdot f(N)$ equals $K' \cdot f(N)$. □

From the previous lemma we have that under certain conditions arc reversal does not influence the quality of the network structure given the data if an information criterion is used. These condition are satisfied as long as arc reversal does not change the independency model represented by the structure. The following lemma now states that any two network structures that represent the same independency model can be transformed into one another by applying successive arc-reversal operations.

4.4 **Lemma** Let V be a set of variables. Let B_S and $B_{S'}$ be network structures over V such that $B_S \equiv B_{S'}$. Then, a finite sequence $B_1, \ldots, B_k$, $k \geq 1$, of network structures over V exists such that $B_S = B_1$, $B_{S'} = B_k$, and for $1 \leq i < k$, $B_{i+1} = arcr(B_i, v_{a_i}, v_{b_i})$ where v_{a_i} and v_{b_i} are nodes in V such that either $v_{a_i} \notin \pi_{b_i}$ or $v_{a_i} \in \pi_{b_i}$ and $\pi_{a_i} = \pi_{b_i} \backslash v_{a_i}$. □

Proof: We prove the lemma by induction on the number of reversed arcs in B_S with respect to $B_{S'}$, that is, the number of pairs of nodes v_i, v_j such that $v_i \rightarrow v_j$ is an arc in B_S and $v_j \rightarrow v_i$ is an arc in $B_{S'}$. For each such a reversed arc, an arc reversal is applied, so if there are k reversed arcs, the sequence is of length k.

If there are zero reversed arcs then B_S is equal to $B_{S'}$ and the lemma is trivially true for $k = 1$.

Assume that the lemma holds for some $k \Leftrightarrow 1 \geq 0$ reversed arcs. Now, let $B_{S'}$ be a network structure that contains k reversed arcs with respect to B_S. We show that an arc reversal on $B_1 = B_S$ can be performed such that the $B_{S'}$ contains $k \Leftrightarrow 1$ reversed arcs with respect to the obtained network structure B_2.

By inspection of the definition of the arc-reversal operation we find that applying the operation to two nodes v_{a_i} and v_{b_i} in B_i for which the conditions of the lemma apply (that is, either $v_{a_i} \notin \pi_{b_i}$ or $v_{a_i} \in \pi_{b_i}$ and $\pi_{a_i} = \pi_{b_i} \backslash v_{a_i}$) results in a network structure B_{i+1} that is equivalent with B_i. So, the independency model of B_{i+1} is the same as the one of B_i if we can find two such nodes v_a and v_b and apply the arc-reversal operation.

Now, let $<_V$ be a topological ordering obeyed of B_1. Let v_a and v_b be two adjacent nodes in B_i on which an arc reversal can be performed but that need not necessarily satisfy the condition in the theorem and let $v_b \rightarrow v_a$ be an arc in $B_{S'}$. Furthermore, let v_b be the lowest ordered node according to $<_V$ for which this property holds.

Now, suppose that $v_a \in \pi_b$ and not $\pi_a = \pi_b \backslash v_a$. We distinguish between two cases for π_a and π_b:

- Suppose that $\pi_b \backslash (\pi_a \cup v_a) \neq \emptyset$. Let u be a node in $\pi_b \backslash (\pi_a \cup v_a)$. Then, we have that $v_a \not\Leftrightarrow u$ holds in the embedded graph of B_1, and that $v_a \rightarrow v_b$ and $v_b \leftarrow u$ are in B_1. If $v_a \rightarrow v_b$ is reversed in $B_{S'}$, then the conditions that $v_a \not\Leftrightarrow u$ is in the underlying graph of B_S and $v_a \rightarrow v_b$ and $v_b \leftarrow u$ are arcs in B_S cannot hold. So, $\pi_b \backslash (\pi_a \cup v_a)$ must be empty to satisfy $B_S \equiv B_{S'}$.

- Suppose that $(\pi_a \cup v_a) \backslash \pi_b \neq \emptyset$. Let u be a node in $(\pi_a \cup v_a) \backslash \pi_b$. Then in B_1, we have $v_a \Leftrightarrow u$, $v_a \Leftrightarrow v_b$ and $u \not\Leftrightarrow v_b$ in the underlying graph of B_1. Furthermore, $\langle v_b, \pi_b, u \rangle$ holds in B_1, thus also in B_S and $B_{S'}$. In B_2 we must have these properties also. However, if the arrow $v_a \rightarrow v_b$ is just flipped in direction, we would obtain $u \rightarrow v_a \leftarrow v_b$ and $\langle v_b, \pi_b, u \rangle$ would not hold. For the d-separation statement to hold in $B_{S'}$, $u \rightarrow v_a$ will have to be flipped in direction in $B_{S'}$ also. But, then a pair v'_a, v'_b would have to exist with $v'_b <_V v_b$ which must be false by our choice of v_b.

From these contradictions, we conclude that either $v_a \notin \pi_b$ or $\pi_a = \pi_b \backslash v_a$.

So, in B_1, always two nodes v_a and v_b can be found to which the arc-reversal operation can be applied such that the represented independency model is not changed. Furthermore, in B_2 there are only $k \Leftrightarrow 1$ reversed arcs with respect to $B_{S'}$, hence by the induction hypothesis there is a finite sequence $B_2, \ldots, B_{S'}$ such that $B_{i+1} = arcr(B_i, v_{a_i}, v_{b_i})$ under the conditions stated in the Lemma. Therefore, there is a finite sequence of network structures $B_S = B_1, \ldots, B_k = B_{S'}$ such that $B_{i+1} = arcr(B_i, v_{a_i}, v_{b_i})$ where v_{a_i} and v_{b_i} are nodes in V such that either $v_{a_i} \notin \pi_{b_i}$ or $v_{a_i} \in \pi_{b_i}$ and $\pi_{a_i} = \pi_{b_i} \backslash v_{a_i}$. □

From the lemma we have that for any equivalent two network structures B_S and $B_{S'}$, a sequence of arc reversals exists that transforms B_S into $B_{S'}$ such that any intermediate network structure is equivalent with B_S and $B_{S'}$. Now combining this result with Lemma 4.3 yields the following theorem.

4.2 **Theorem** Let V be a set of variables. Let D be a database over V. Let the prior probability distribution on network structures over V be uniform. Let I_f be an information criterion with penalty function f. Then,

$$I_f(B_S, D) = I_f(B_{S'}, D),$$

for any two network structures B_S and $B_{S'}$ with $B_S \equiv B_{S'}$. □

Proof: Let B_S and $B_{S'}$ be network structures with $B_S \equiv B_{S'}$. From Lemma 4.4, we have that a finite sequence of network structures $B_1, \ldots, B_k$, $k \geq 1$, exists such that $B_S = B_1$, $B_{S'} = B_k$, and $B_{i+1} = arcr(B_i, v_{a_i}, v_{b_i})$ with either $v_{a_i} \notin \pi_{b_i}$ or $v_{a_i} \in \pi_{b_i}$ and $\pi_{a_i} = \pi_{b_i} \backslash v_{a_i}$ in B_i. From Lemma 4.3, we have that such arc reversals do not change the quality of the resulting network structures according to an information criterion. So, $I_f(B_{i+1}, D) = I_f(B_i, D)$ for $i = 1, \ldots, k$ hence $I_f(B_S, D) = I_f(B_{S'}, D)$. □

From this theorem we have that the information criteria are score equivalent if a uniform prior distribution over the network structures is assumed, as opposed to the Bayesian measure.

4.2.6 Infinite-Size Database Properties of Quality Measures

In this section, we will investigate the asymptotic behavior of the Bayesian measure, information criteria, and the MDL measure, that is, the behavior for infinite-size databases. Although the results will be of a theoretical nature mainly, they indicate how the measures may be expected to behave for large databases.

For the Bayesian measure it is known that it assigns a higher quality to minimal I-maps than to any other network structures given a large database obeying the same topological ordering as the I-map when the database is generated from a positive distribution [57]. Since the MDL measure approximates the Bayesian measure, a similar property holds for the MDL measure. In fact, the property also holds for information criteria as well and can be generalized even more, as is stated in the following theorem.

4.3 **Theorem** Let V be a set of variables and let $<_V$ a total ordering one V. Let the prior probability distribution over all network structures over V be positive. Now, let Pr_D be a joint probability distribution over V such that B_S is a minimal I-map of Pr_D obeying $<_V$ and no other network structure obeying $<_V$ is a minimal I-map of Pr_D. Let D be a database with N cases generated from Pr_D. Let Q be either the Bayesian measure, the MDL measure, or an information criterion with nonzero penalty function f, where $\lim_{N\to\infty} f(N) = \infty$ and $\lim_{N\to\infty} f(N)/N = 0$. Then, for any network structure $B_{S'}$ over V that obeys $<_V$, we have that

$$\lim_{N\to\infty} (Q(B_{S'}, D) \Leftrightarrow Q(B_S, D)) = \Leftrightarrow\infty$$

if and only if $B_{S'}$ is not a minimal I-map of Pr_D. □

Proof: For the Bayesian measure, the property stated above follows from Theorem 6.3 of [57]. Since the MDL measure is a special case of an information criterion that satisfies the above conditions, it suffices to prove the theorem for information criteria. Let f be a penalty function such that $\lim_{N\to\infty} f(N) = \infty$ and $\lim_{N\to\infty} f(N)/N = 0$ and consider information criterion I_f.

Let N, n, r_i, v_i, q_i, x_{ik}, $x_{\pi_i j}$, N_{ijk}, and N_{ij} be as before for the network structure B_S and database D, and let r'_i, v'_i, q'_i, x'_{ik}, $x'_{\pi_i j}$, N'_{ijk}, and N'_{ij} be likewise for $B_{S'}$ and D. Note that $r'_i = r_i$ for $i = 1, \ldots, n$. Let K and K' be the numbers of parameters for B_S and $B_{S'}$, respectively. We consider $\lim_{N\to\infty} (I_f(B_{S'}, D) \Leftrightarrow I_f(B_S, D))$ which by definition is equal to

4.16 $$\lim_{N\to\infty} (\log P(B_{S'}) \Leftrightarrow NH(B_{S'},D) \Leftrightarrow K'f(N) \Leftrightarrow \log P(B_S) + NH(B_S,D) + Kf(N))\,.$$

Now, consider the behavior of the entropy term $N \cdot H(B_S, D)$ in the limit for $N \to \infty$. This term is by definition $\Leftrightarrow N \cdot \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \Leftrightarrow \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}}$. By the strong law of large numbers, we have that $\lim_{N\to\infty} \frac{N_{ijk}}{N} = Pr_D(v_i = x_{ik}, \pi_i = x_{\pi_i j})$ and $\lim_{N\to\infty} \frac{N_{ijk}}{N_{ij}} = Pr_D(v_i = x_{ik}|\pi_i = x_{\pi_i j})$. Therefore, in the limit for $N \to \infty$ we have that

$$N \cdot H(B_S, D) = N \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \Leftrightarrow Pr_D(v_i = x_{ik}, \pi_i = x_{\pi_i j}) \log Pr_D(v_i = x_{ik}|\pi_i = x_{\pi_i j}).$$

A similar property holds for the behavior of the entropy term $N \cdot H(B_{S'}, D)$. To examine the behavior of the sum $N(\Leftrightarrow H(B_{S'}, D) + \cdot H(B_S, D))$ in Expression 4.16, we distinguish between three cases:

- $B_{S'}$ is not an I-map of Pr_D,
- $B_{S'}$ is an I-map of Pr_D but not a minimal one, and
- $B_{S'}$ is a minimal I-map of Pr_D.

First, suppose that $B_{S'}$ is not an I-map of Pr_D. Then, in the limit for $N \to \infty$ the sum $\Leftrightarrow N \cdot H(B_{S'}, D) + N \cdot H(B_S, D)$ is equal to

$$N \cdot \sum_{i=1}^{n} \left\{ \sum_{j=1}^{q'_i} \sum_{k=1}^{r'_i} Pr_D(v_i = x_{ik}, \pi'_i = x'_{\pi_i j}) \log Pr_D(v_i = x_{ik}|\pi'_i = x'_{\pi_i j}) \right.$$
$$\left. \Leftrightarrow \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} Pr_D(v_i = x_{ik}, \pi_i = x_{\pi_i j}) \log Pr_D(v_i = x_{ik}|\pi_i = x_{\pi_i j}) \right\}.$$

Now observe that since $B_{S'}$ is not an I-map of Pr_D, there is an index i such that $\pi_i \not\subseteq \pi'_i$; if for all i we would have $\pi_i \subseteq \pi'_i$, then $B_{S'}$ would represent less independencies than the I-map B_S and $B_{S'}$ would be an I-map too. For this index i, let $\pi''_i = \pi_i \pi'_i$, let $x''_{\pi_i j}$ be the jth configuration of π''_i, and let q''_i be the number of all possible configurations of π''_i. Now observe that for $j = 1, \ldots, q''_i$, we have that $Pr_D(v_i = x_{ik}|\pi''_i = x''_{\pi_i j}) = Pr_D(v_i = x_{ik}|\pi_i = x_{\pi_i})$ where x_{π_i} conforms to $x''_{\pi_i j}$, because v_i and π''_i are conditionally independent given π_i, that is, $I(v_i, \pi_i, \pi''_i)$. So, by marginalization the above equation can be written as,

$$N \cdot \sum_{i=1}^{n} \left\{ \sum_{j=1}^{q''_i} \sum_{k=1}^{r_i} Pr_D(v_i = x_{ik}, \pi''_i = x''_{\pi_i j}) \log Pr_D(v_i = x_{ik}|\pi'_i = x'_{\pi_i}) \right.$$
$$\left. \Leftrightarrow \sum_{j=1}^{q''_i} \sum_{k=1}^{r_i} Pr_D(v_i = x_{ik}, \pi''_i = x''_{\pi_i j}) \log Pr_D(v_i = x_{ik}|\pi''_i = x''_{\pi_i j}) \right\},$$

where x'_{π_i} conforms to $x''_{\pi_i j}$. From Shannon's inequality, which states $\sum_i \Leftrightarrow a_i \log a_i \leq \sum_i \Leftrightarrow a_i \log b_i$ for all $a_i, b_i \geq 0$ such that $\sum_i a_i = \sum_i b_i = 1$, we have that the expression within brackets must be greater than or equal to 0 because there is at least one index i for which $Pr_D(v_i = x_{ik}|\pi'_i = x_{\pi_i})$ is not equal to $Pr_D(v_i = x_{ik}|\pi''_i = x''_{\pi_{ij}})$. So, in the limit for $N \to \infty$ the entropy of $B_{S'}$ is higher than the entropy of B_S. Note that for the other two cases the Bayesian belief networks represent the same probability distribution, namely Pr_D. So, for those cases we have $N(\Leftrightarrow H(B_{S'}, D) + \cdot H(B_S, D)) = 0$.

To examine the behavior of Expression 4.16, we distinguish between the same three cases as before.

If $B_{S'}$ is not an I-map of Pr_D, the entropy terms sum to $\Leftrightarrow\infty$ when $N \to \infty$. Since $O(N)$ terms dominate $O(f(N))$ when $N \to \infty$ we have that the term $(K \Leftrightarrow K') \cdot f(N)$ in Expression 4.16 is dominated by the term $\Leftrightarrow N \cdot H(B_{S'}, D) + N \cdot H(B_S, D) \to \Leftrightarrow\infty$. Since the prior probability distribution on network structures is positive, the term $\log(P(B_{S'})/P(B_S))$ is a constant with respect to to N and negligible in the limit for $N \to \infty$. So, $\lim_{N\to\infty} (Q(B_{S'}, D) \Leftrightarrow Q(B_S, D)) = \Leftrightarrow\infty$.

If $B_{S'}$ is a non-minimal I-map of Pr_D, $B_{S'}$ must comprise at least one extra arc compared to $B_{S'}$. So, $K' \Leftrightarrow K > 0$. Since for the penalty function the property $\lim_{N\to\infty} f(N) = \infty$ holds, we have that, $\Leftrightarrow(K' \Leftrightarrow K)f(N) \to \Leftrightarrow\infty$. The term $\log(P(B_S)/P(B_{S'}))$ can once more be neglected and therefore, $\lim_{N\to\infty}$ $(Q(B_{S'}, D) \Leftrightarrow Q(B_S, D)) = \Leftrightarrow\infty$.

If $B_{S'}$ is a minimal I-map then $B_{S'}$ equals B_S since it was required that the minimal I-map is unique for the given ordering and $\lim_{N\to\infty} (Q(B_{S'}, D) \Leftrightarrow Q(B_S, D)) = 0$. □

The theorem states that for large enough databases, a network structure that is a minimal I-map obeying a particular ordering $<_V$ is overwhelmingly preferred over any other network structure obeying the same ordering. From the fact that independency models associated with positive distributions are closed under the graphoid axioms and that such independency models have a unique minimal I-maps for network structures that obey a given ordering, this property holds for any positive distribution.

Equally, the property holds for any distribution that has a P-map, since independency models associated with P-maps are closed under the graphoid axioms as well. So, for these distributions also there is a unique minimal I-map for every given ordering. In general for finite-size databases, a minimal I-map need not necessarily have a higher quality than other structures.

Note that application of Theorem 4.3 requires that an ordering on the variables is given. For learning a network structure, however, it is not desirable that an ordering on the variables need to be provided to the learning algorithm; different orderings may results in considerably different network structures and finding a 'good' ordering may be difficult. Therefore, we investigate the behavior of the quality measures more in general. We are interested in determining the class of network structures that are preferred over other network structures by the considered quality measures in order to get a better understanding of the quality measures.

We begin by observing that minimal I-maps need not be unique. In fact, different minimal I-maps for a joint probability distribution need not even be equivalent. Consider for example the network structures shown in Figure 4.3. Suppose that the structure on the left is a P-map of some distribution Pr. Both structures on the right are minimal I-maps of the distribution obtained from Pr by marginalizing

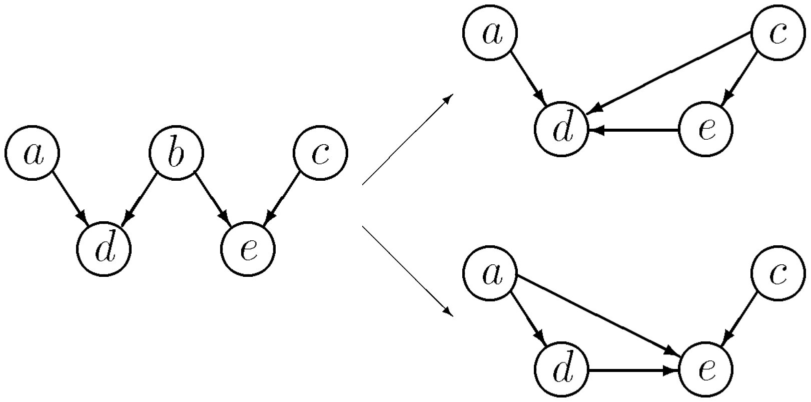


Figure 4.3: Example of different minimal I-maps.

over b; note that the two structures do not obey the same variable ordering. The two I-maps are not equivalent; for example, the upper structure represents $I(a, \emptyset, e)$ whereas the lower structure does not.

As a result of the fact that minimal I-maps are not unique, the numbers of parameters that need to be estimated for different minimal I-maps need not necessarily the same. Therefore, a quality measure may assign different qualities to different minimal I-maps. It is desirable that as few as possible probabilities need be assessed, since in every estimate of a probability a small error is introduced. This motivates distinguishing between minimum and non-minimum I-maps.

4.7 **Definition** Let V be a set of variables. Let M^I be an independency model over V and B_S be a network structure over V. Then, B_S is a *minimum I-map* of M^I if B_S is an I-map of M^I and for every network structure $B_{S'}$ over V that is an I-map of M^I, we have that $K' \geq K$, where K and K' are the number of parameters of B_S and $B_{S'}$, respectively. □

Let Pr be a joint probability distribution over V. Then, we say that B_S is a minimum I-map of Pr to denote that B_S is a minimum I-map of the independency model M^I that is associated with Pr.

Consider once more the network structures on the left-hand side in Figure 4.3. Suppose that all variables except c are binary and that the variable is c ternary. Then, for the upper structure on the right of Figure 4.3 eighteen probabilities need be specified to arrive at a belief network: one probability for a, two for c, twelve for d, and three for e. However, for the lower structure, only seventeen probabilities need be specified: one for a, two for c, two for d, and twelve for e.

Unfortunately, in general minimum I-maps need not be unique. Further, two minimum I-maps may exits that are not equivalent. Consider the network structures on the left-hand side in Figure 4.3 once more. If all variables including c would be binary, for both structures twelve probabilities would need to be specified.

However, the quality measures that we consider all prefer minimum I-maps over non-minimum I-maps under some conditions stated in the following theorem.

4.4 **Theorem** Let V be a set of variables and let the prior probability distribution over all network structures over V be positive. Let Pr_D be a positive joint probability distribution over V, and let B_S be a minimum I-map of Pr_D. Let D be a database with N cases generated from Pr_D. Let Q be either the Bayesian measure, the MDL measure, or an information criterion with nonzero penalty function f where $\lim_{N\to\infty} f(N) = \infty$ and $\lim_{N\to\infty} f(N)/N = 0$. Then, for any network structure $B_{S'}$ over V, we have that

$$\lim_{N\to\infty} (Q(B_S, D) \Leftrightarrow Q(B_{S'}, D)) = \Leftrightarrow\infty$$

if and only if $B_{S'}$ is not a minimum I-map of Pr_D. □

Proof: Recall from Theorem 4.1 that for large enough N the MDL measure is an approximation of the Bayesian measure with $O(1)$ error provided that each configuration of every parent set occurs in the database. Since Pr_D is a positive distribution, all these configurations are guaranteed to be in the database for $N \to \infty$. Also observe that for penalty function $f(N) = \frac{1}{2}\log N$, the information criterion $I_{\frac{1}{2}\log}$ is equal to the MDL measure. We conclude that it suffices to prove the theorem for the information criteria in general. Let f be a penalty function such that $\lim_{N\to\infty} f(N) = \infty$ and $\lim_{N\to\infty} f(N)/N = 0$ and consider the information criterion I_f.

Let N, n, r_i, v_i, q_i, x_{ik}, $x_{\pi_{ij}}$, N_{ijk}, and N_{ij} be as before for the network structure B_S and database D, and let r'_i, v'_i, q'_i, x'_{ik}, $x'_{\pi_{ij}}$, N'_{ijk}, and N'_{ij} be likewise for $B_{S'}$ and D. Let K and K' be the numbers of parameters for B_S and $B_{S'}$, respectively. We consider the expression $\lim_{N\to\infty} (I_f(B_{S'}, D) \Leftrightarrow I_f(B_S, D))$, which by definition is equal to $\lim_{N\to\infty} (\log P(B_{S'}) \Leftrightarrow NH(B_{S'}, D) \Leftrightarrow K'f(N) \Leftrightarrow \log P(B_S) + NH(B_S, D) + Kf(N))$. This expression can be written as

4.17 $$\lim_{N\to\infty} \left(\log \frac{P(B_{S'})}{P(B_S)} \Leftrightarrow N \cdot H(B_{S'}, D) + N \cdot H(B_S, D) \Leftrightarrow (K' \Leftrightarrow K) \cdot f(N) \right).$$

First, consider the entropy term $H(B_{S'}, D)$ and its behavior in the limit for $N \to \infty$. This term by definition is $\Leftrightarrow\sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \Leftrightarrow \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}}$. By the strong law of large numbers, we have that $\lim_{N\to\infty} \frac{N'_{ijk}}{N} = Pr_D(v_i = x'_{ik}, \pi'_i = x'_{\pi_{ij}})$ and $\lim_{N\to\infty} \frac{N'_{ijk}}{N'_{ij}} = Pr_D(v_i = x'_{ik} | \pi'_i = x'_{\pi_{ij}})$. Therefore in the limit for $\lim_{N\to\infty}$ we have that $H(B_{S'}, D)$ can be written as

$$\sum_{i=1}^{n} \sum_{j=1}^{q'_i} \sum_{k=1}^{r'_i} \Leftrightarrow Pr_D(v_i = x'_{ik}, \pi'_i = x'_{\pi_{ij}}) \cdot \log Pr_D(v_i = x'_{ik} | \pi'_i = x'_{\pi_{ij}}).$$

By renaming configurations, this is equal to

$$\sum_{i=1}^{n} \sum_{x_{v_i} \in \Omega_i} \sum_{x_{\pi'_i} \in \Omega_{\pi'_i}} \Leftrightarrow Pr_D(v_i = x_{v_i}, \pi'_i = x_{\pi'_i}) \cdot \log Pr_D(v_i = x_{u_i} | \pi'_i = x_{\pi'_i}).$$

By marginalization we have

$$\sum_{i=1}^{n} \sum_{x_V \in \Omega_V} \Leftrightarrow Pr_D(V = x_V) \cdot \log Pr_D(v_i = x_{u_i} | \pi'_i = x_{\pi'_i}).$$

Now, let $<_V$ be an ordering obeyed by $B_{S'}$ and let $V_i = \{v | v <_V v_i\}$ for $i = 1, \ldots, n$. Then, we can rewrite the above expression as

$$\sum_{i=1}^{n} \sum_{x_V \in \Omega_V} \Leftrightarrow \left(\prod_{h=1}^{n} Pr_D(u_h = x_h | V_h = x_{V_h}) \right) \cdot \log Pr_D(v_i = x_{u_i} | \pi'_i = x_{\pi'_i}).$$

Changing the order of summation and collection the summation over i in the logarithm gives

$$\sum_{x_V \in \Omega_V} \Leftrightarrow \left(\prod_{i=1}^{n} Pr_D(u_i = x_i | V_i = x_{V_i}) \right) \cdot \log \left(\prod_{i=1}^{n} Pr_D(v_i = x_{u_i} | \pi'_i = x_{\pi'_i}) \right).$$

Now, let $Pr'_D(V = x_V) = \prod_{i=1}^{n} Pr_D(v_i = x_{v_i} | \pi'_i = x_{\pi'_i})$, that is, let Pr'_D be the joint probability distribution defined by the belief network with network structure $B_{S'}$. Then, we can write the above expression as

$$\sum_{x_V \in \Omega_V} \Leftrightarrow Pr_D(V = x_V) \cdot \log Pr'_D(V = x_V).$$

We now consider the entropy term $H(B_S, D)$. By a similar argument, we find that this term equals

$$\sum_{x_V \in \Omega_V} \Leftrightarrow Pr_D(V = x_V) \cdot \log Pr_D(V = x_V).$$

Note that B_S is an I-map of Pr_D. Therefore, the distribution represented by the Bayesian belief network with B_S is equal to Pr_D and the expression for $H(B_S, D)$ contains the same distribution before as after the logarithm. So, Expression 4.17 can be written as

$$\lim_{N \to \infty} \left(\log \frac{P(B_{S'})}{P(B_S)} + N \cdot \sum_{x_V \in \Omega_V} (Pr_D(V = x_V) \cdot \log Pr'_D(V = x_V) \right.$$

$$\left. \Leftrightarrow Pr_D(V = x_V) \cdot \log Pr_D(V = x_V)) \Leftrightarrow (K' \Leftrightarrow K) \cdot f(N) \right). \tag{4.18}$$

We now distinguish between three cases:

- $B_{S'}$ is not an I-map of Pr_D,
- $B_{S'}$ is an I-map but not a minimum I-map of Pr_D, and
- $B_{S'}$ is a minimum I-map of Pr_D.

We consider these cases separately. First, suppose that $B_{S'}$ is not an I-map of Pr_D. Then, there is a configuration x_V of V such that $Pr'_D(V = x_V) \neq Pr_D(V = x_V)$; if no such configuration would exist, then the belief networks with the network structure $B_{S'}$ and B_S would define the same probability distribution and $B_{S'}$ would model at most the same set of independency statements as the I-map B_S and therefore would be an I-map too. From Shannon's inequality, which states $\sum_i a_i \log b_i \leq \sum_i a_i \log a_i$, for all $a_i, b_i \geq 0$ such that $\sum_i a_i = \sum_i b_i = 1$, we have that the sum $\sum_{x_V \in \Omega_V} (Pr_D(V = x_V) \cdot \log Pr'_D(V = x_V) \Leftrightarrow Pr_D(V = x_V) \cdot \log Pr_D(V = x_V))$ in Expression 4.18 is less than 0. The result is multiplied by N yielding a negative $O(N)$

term. Since the prior probability distribution on network structures is positive, the term $\log(P(B_{S'})/P(B_S))$ is a constant with respect to N and negligible in the limit for $N \rightarrow \infty$. Since an $O(N)$ term dominates any $O(f(N))$ term for $N \rightarrow \infty$ we have that the $(K' \Leftrightarrow K) \cdot f(N)$ in Expression 4.18 is dominated by the term $N \cdot \sum_{x_V \in \Omega_V} (Pr_D(V = x_V) \cdot \log Pr'_D(V = x_V) \Leftrightarrow Pr_D(V = x_V) \cdot \log Pr_D(V = x_V))$. So, Expression 4.18 equals $\Leftrightarrow\infty$.

Now suppose that $B_{S'}$ is an I-map of Pr_D, yet not a minimum I-map. We observe that the joint probability distribution defined by the Bayesian belief network with $B_{S'}$ equals Pr_D and that the entropy terms in Expression 4.18 cancel out. However, since $B_{S'}$ is not a minimum I-map of Pr_D, we know that the number of parameters of $B_{S'}$ is larger than number of parameters of B_S. Therefore, $K' \Leftrightarrow K > 0$. So, since for penalty function f the property $\lim_{N\rightarrow\infty} f(N) = \infty$ holds, we have that the term $\Leftrightarrow(K' \Leftrightarrow K) \cdot f(N) \rightarrow \Leftrightarrow\infty$ for $N \rightarrow \infty$. The term $\log P(B_S)/P(B_{S'})$ can once more be neglected. So, Expression 4.18 equals $\Leftrightarrow\infty$.

To conclude, suppose that $B_{S'}$ is a minimum I-map of Pr_D. Then, following the same line of reasoning above, we find that $H(B_S, D) = H(B_{S'}, D)$ and further that $K = K'$. Expression 4.18 then equals $\log P(B_S)/P(B_{S'})$ which is a finite constant if the prior probability distribution on network structures is positive. □

The theorem states that for large enough databases, a network structure that is a minimum I-map is assigned a far larger quality than any other network structure. Note that the property stated in the theorem is more general than the property stated in Theorem 4.3 as it does not depend on an ordering of the variables. Now if a probability distribution has a P-map, then for every ordering there is a unique P-map, and by Theorem 4.3 we have this P-map is assigned a far higher quality than any other network structure. In general however, P-maps need not be unique. But, we have the following property.

4.5 **Theorem** Let V be a set of variables. Let the prior probability distribution over all network structures over V be positive. Let Pr_D be a positive distribution over V such that a P-map exists for Pr_D. Let B_S be such a P-map for Pr_D. Now, let D be a database with N cases generated from Pr_D. Let Q be either the Bayesian measure, the MDL measure, or an information criterion with nonzero penalty function f such that $\lim_{N\rightarrow\infty} f(N) = \infty$ and $\lim_{N\rightarrow\infty} f(N)/N = 0$. Then, for any network structure $B_{S'}$ over V we have that

$$\lim_{N\rightarrow\infty} Q(B_S, D) \Leftrightarrow Q(B_{S'}, D) = \Leftrightarrow\infty$$

if and only if $B_{S'}$ is not a P-map of Pr_D. □

Proof: This proof is similar to the proof of Theorem 4.4. Following a same line of reasoning as in the proof of Theorem 4.4, we find that we have to show that

4.19
$$\lim_{N\rightarrow\infty} \left(\log \frac{P(B_S)}{P(B_{S'})} \Leftrightarrow N \cdot H(B_{S'}, D) + N \cdot H(B_S, D) \Leftrightarrow (K' \Leftrightarrow K) \cdot f(N) \right)$$

goes to minus infinity if and only if $B_{S'}$ is not a minimal I-map. We distinguish three cases:

- $B_{S'}$ is not a minimal I-map,
- $B_{S'}$ is a minimal I-map but not a P-map, and
- $B_{S'}$ is a P-map.

If $B_{S'}$ is not a minimal I-map of Pr_D, the corollary follows from the observation that a P-map is a minimal I-map. From [73] we have the following property; let $<_V$ be an ordering obeyed by B_{S_1}, then a sequence of arc reversals on B_{S_1} exists that results in network structure B_{S_2} such that B_{S_2} is a minimal I-map of the independency model represented by B_{S_1}. So, a sequence of arc reversals exists on B_S such that the resulting network structure $B_{S''}$ obeys a same topological ordering as $B_{S'}$ and $B_{S''}$ is a minimal I-map of the independency model represented by B_S. Since B_S is a P-map of Pr_D, we conclude that $B_{S''}$ is a minimal I-map of Pr_D. Since a P-map exists, we conclude that Pr_D satisfies intersection and thus that minimal I-maps are unique for a given ordering and Theorem 4.3 applies.

If $B_{S'}$ is a minimal I-map but not a P-map, let $B_{S''}$ as before. Since both $B_{S'}$ and $B_{S''}$ are minimal I-maps and minimal I-maps are unique for a given ordering, $B_{S''}$ must be equal to $B_{S'}$. With every arc reversal, zero or more arcs are added. Assume that during the sequence of arc reversal no arcs are added. Then $B_{S'}$ represents the same set of independency statements as B_S and thus $B_{S'}$ is a P-map. This is not true by our primary assumption, so we conclude that an arc must have been added during the sequence of arc reversals and the number of parameters for $B_{S'}$ is larger than the number of parameters for B_S, that is $K' > K$. Since $B_{S'}$ is an I-map, $H(B_{S'}, D) = H(B_S, D)$. The term $\log \frac{P(B_S)}{P(B_{S'})}$ is a constant that vanishes in Formula 4.19. Therefore, Formula 4.19 will go to $\Leftrightarrow\infty$.

If $B_{S'}$ is a P-map, then a sequence of arc reversals exists that transforms B_S into $B_{S'}$ such that no arcs are added during the arc reversals by Lemma 4.4. So, B_S and $B_{S'}$ induce the same number of parameters and represent the same distributions. And we conclude that Formula 4.19 is a constant $\log \frac{P(B_S)}{P(B_{S'})}$ that is finite because the prior on network structures was assumed to be positive. □

Theorem 4.5 states that for large enough databases, a network structure that is a perfect map is assigned a far larger quality than other network structures. This is an important property when one is interested in learning causal structure as performed by Spirtes [104]; for conditional independence based causal structure recovery it is assumed that there exists a perfect map, which is the so-called 'faithfulness condition'. So, by Theorem 4.5 it is justified that the quality measures considered are used for detecting causal relations.

4.2.7 Finite-Size Database Properties of Quality Measures

In the previous section, the various quality measures have been compared with respect to their behavior for databases of infinite size. Since for real-life applications infinite-size databases never occur, we also are interested in the non-asymptotic behavior of these measures. It can be expected that this behavior differs from the behavior for infinite-size databases because the estimates of the probabilities in the entropy terms will contain an error. Further, the Bayesian measure and MDL measure may differ a lot because the condition under which they are approximately the same need not necessarily hold. Now recall that one of the aims of learning is to construct a Bayesian belief network for use as inference in knowledge-based systems. As the complexity of belief network inference often is determined by the size of the parent sets of a network, it is interesting to investigate the behavior of the various quality measures with respect to the parent-set size. Insight in this behavior further helps in investigating search algorithms. The following theorem gives some insight on the behavior of the information criterion.

4.6 **Theorem** Let V be a set of variables. Let the prior probability distribution over all network structures over V be uniform. Let D be a database with N cases over V. Let B_S be a network structure over V with a parent set containing more than $\log\left(\frac{N}{f(N)}+1\right)$ variables. Then, a network structure $B_{S'}$ exists such that $I_f(B_{S'},D) > I_f(B_S,D)$ with fewer arcs, where I_f be an information criterion with non-zero penalty function f. □

Proof: Let v_s be a variable in V such that $|\pi_s| > \log\left(\frac{N}{f(N)}+1\right)$ in B_S. Now, let $B_{S'}$ be the network structure obtained from B_S by deleting all incoming arcs for this variable v_s, that is, $\pi'_s = \emptyset$. Let r_i, q_i, N_{ij}, and N_{ijk} be as before for B_S and D and r'_i, q'_i, N'_{ij} and N'_{ijk} likewise for $B_{S'}$ and D; note that $r_i = r'_i$, $i = 1,\ldots,n$. We prove the theorem by contradiction. Suppose that $B_{S'}$ is not assigned a higher quality according to I_f than B_S. We consider the difference $I_f(B_{S'},D) \Leftrightarrow I_f(B_S,D)$, which equals

$$\log\frac{P(B_{S'})}{P(B_S)} \Leftrightarrow N\cdot(H(B_{S'},D)\Leftrightarrow H(B_S,D))\Leftrightarrow(K'\Leftrightarrow K)\cdot f(N). \qquad 4.20$$

Since $B_{S'}$ is not assigned a higher quality than B_S, this difference is not positive. Because the prior probability distribution over all network structures is uniform, we have that $\log(P(B_{S'})/P(B_S)) = 0$.

Now consider the entropy terms in expression 4.20. By definition of entropy, we have that $H(B_{S'},D)\Leftrightarrow H(B_S,D)$ equals

$$\sum_{i=1}^{n}\sum_{j=1}^{q'_i}\sum_{k=1}^{r'_i}\Leftrightarrow\frac{N'_{ijk}}{N}\log\frac{N'_{ijk}}{N'_{ij}}\Leftrightarrow\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}\Leftrightarrow\frac{N_{ijk}}{N}\log\frac{N_{ijk}}{N_{ij}}.$$

We observe that for all $i \neq s$, the separate terms cancel out. So, we can write $H(B_{S'},D)\Leftrightarrow H(B_S,D)$ as

$$\sum_{j=1}^{q'_s}\sum_{k=1}^{r_s}\Leftrightarrow\frac{N'_{sjk}}{N}\log\frac{N'_{sjk}}{N'_{sj}}\Leftrightarrow\sum_{j=1}^{q_s}\sum_{k=1}^{r_s}\Leftrightarrow\frac{N_{sjk}}{N}\log\frac{N_{sjk}}{N_{sj}},$$

which equals

$$\sum_{j=1}^{q'_s}\frac{N'_{sj}}{N}\left(\sum_{k=1}^{r_s}\Leftrightarrow\frac{N'_{sjk}}{N'_{sj}}\log\frac{N'_{sjk}}{N'_{sj}}\right)\Leftrightarrow\sum_{j=1}^{q_s}\frac{N_{sj}}{N}\left(\sum_{k=1}^{r_s}\Leftrightarrow\frac{N_{sjk}}{N_{sj}}\log\frac{N_{sjk}}{N_{sj}}\right).$$

We maximize this expression to compensate for the penalty term in Expression 4.20. To this end, we use the property that $0 \leq \sum_{k=1}^{r} \Leftrightarrow p_k \log p_k \leq \log r$, for $p_k \geq 0$, $k = 1, \ldots, r$ with $\sum_{k=1}^{r} p_k = 1$ for $H(B_{S'}, D) \Leftrightarrow H(B_S, D)$ we find

$$H(B_{S'}, D) \Leftrightarrow H(B_S, D) \leq \sum_{j=1}^{q'_s} \frac{N'_{sj}}{N} \cdot \log r_s \Leftrightarrow \sum_{j=1}^{q_s} \frac{N_{sj}}{N} \cdot 0 = \log r_s.$$

The latter equality follows from the observation that $\sum_{j=1}^{q'_s} \frac{N'_{sj}}{N} = 1$. We therefore have that $\Leftrightarrow N \cdot (H(B_{S'}, D) \Leftrightarrow H(B_S, D)) \geq \Leftrightarrow N \cdot \log r_s$. Since Expression 4.20 is not positive, it follows that $\Leftrightarrow (K' \Leftrightarrow K) \cdot f(N) \leq N \cdot \log r_s$. Therefore, the following inequality holds

$$(r_s \Leftrightarrow 1) \cdot q_s \cdot f(N) \Leftrightarrow (r_s \Leftrightarrow 1) \cdot f(N) \leq N \cdot \log r_s.$$

Note that $q'_s = 1$ since $\pi'_s = \emptyset$. Division of this expression by $(r_s \Leftrightarrow 1) \cdot f(N)$, which is admissible since $f(N) > 0$, gives

$$q_s \Leftrightarrow 1 \quad \leq \quad \frac{N}{f(N)} . \frac{\log r_s}{r_s \Leftrightarrow 1}.$$

Using the inequality $\log x \leq (x \Leftrightarrow 1)$ for positive integers x, we find

$$q_s \Leftrightarrow 1 \quad \leq \quad \frac{N}{f(N)}.$$

So,

$$q_s \leq \frac{N}{f(N)} + 1.$$

Now, recall that $q_s = \prod_{v_j \in \pi_s} r_j$, Since for each variable v_j we have $r_j \geq 2$, $j = 1, \ldots, n$ it follows that $q_s \geq 2^{|\pi_s|}$ and hence that $|\pi_s| \leq \log q_s$. So,

$$|\pi_s| \leq \log \left(\frac{N}{f(N)} + 1 \right).$$

This contradicts the number of parents of v_s in B_S being larger than $\log(\frac{N}{f(N)} + 1)$. From this contradiction, we conclude that Expression 4.20 is positive and therefore that $B_{S'}$ is assigned a higher quality according to I_f than B_S. □

Theorem 4.6 implies that good search algorithms that use an information criterion will never select network structures with parent sets with more than $\log \left(\frac{N}{f(N)} + 1 \right)$ variables, when N is the number of cases in the database used for learning. So, with the AIC measure where the penalty function equals $f(N) = 1$, no structures with parent sets with more than $\log (N + 1)$ variables should be found; note that almost equal to $\log N$. With the BIC measure where the penalty function is $f(N) = \frac{1}{2} \log N$, no parent sets with more than $\log \left(\frac{2N}{\log N} + 1 \right)$ variables should be found, which is smaller than $\log N$ for large enough N. Since the BIC criterion and the MDL

v_7	v_6	v_5	v_4	v_3	v_2	v_1	w	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	D_7
1	0	0	0	0	0	0	1	
1	0	0	0	0	0	0	1	D_6
1	1	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	D_5
1	1	1	0	0	0	0	1	
1	1	1	0	0	0	0	1	D_4
1	1	1	1	0	0	0	0	
1	1	1	1	0	0	0	0	D_3
1	1	1	1	1	0	0	1	
1	1	1	1	1	0	0	1	D_2
1	1	1	1	1	1	0	0	
1	1	1	1	1	1	1	1	D_1

Figure 4.4: The databases D_7.

measure coincide, $\log\left(\frac{2N}{\log N}+1\right)$ is also the upper-bound on the size of the parent sets of the network structure that should be found when using the MDL measure. With penalty functions larger than $f(N)=1$, $\log\left(\frac{N}{f(N)}+1\right)$ is smaller than $\log N$ for large enough N. So for a wide range of information criteria, $\log N$ can be taken as an upper-bound of the parent sets in selected network structures.

Since the Bayesian measure and MDL measure are approximately the same for large enough databases, a similar result would be expected for the Bayesian measure. However, the Bayesian measure does not have such a property. Consider a database D_n defined recursively by

$$D_1 = \begin{array}{cc} v_1 & w \\ 0 & 0 \\ 1 & 1 \end{array}$$

where the rows represent the individual cases and the columns indicate the values of the variables; database D_j is constructed from D_{j-1}, $j=2,\ldots,n$ by adding an extra column for a new variable v_j filled with 1s, and two new identical cases with $v_i=0$ $i=1,\ldots,j$ and $w=(j+1)\text{mod }2$. For example, Figure 4.4 shows the database D_7 as it is built from D_1 up to D_6.

For the database D_7, the network structure with highest quality according to the Bayesian measure is shown in Figure 4.5. Note that each node v_i, $i=1,\ldots,7$, has node v_{i-1} in its parent set except for node v_1 which has an empty parent set. Node w includes all nodes $v_1,\ldots,v_7$ in its parent set. This network was found by brute force: the Bayesian measure was calculated for all 1.138.779.265 possible network structures over the variables discerned. Obviously, the parent set of node w contains more than $\log(14)\approx 3.81$ parents.

To explain why the network structure shown in Figure 4.5 is assigned highest quality for the database D_7, we examine the behavior of the Bayesian measure for the database D_n in general. Informally speaking, the Bayesian measure will favor

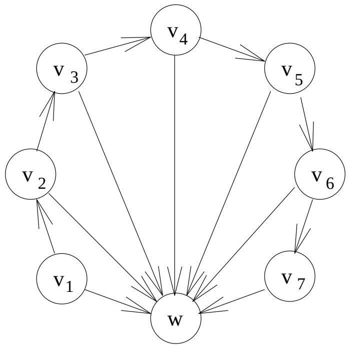


Figure 4.5: The network structure with highest quality according to the Bayesian measure for D_7.

a parent set π_i for a variable v_i over a subset of π_i if knowledge of the values of the variables in π_i yields information about the distribution over the values of v_i, as long as the number of different configurations of π_i that occur in the database is not too large. Now, consider the parent sets of the best structure B_S for D_n according to the Bayesian measure. Note that, relative to D_n, v_i's value is a function of the values of v_{i-1}, v_{i+1} and w. Therefore, π_i will be a subset of $\{v_{i-1}, v_{i+1}, w\}$. In B_S each node v_i, $i = 2, \ldots, n \Leftrightarrow 1$, may have v_{i-1} in its parent set: whenever $v_{i-1} = 1$, we find in the database that $v_i = 1$ and when $v_{i-1} = 0$, we find that $v_i = 0$ for almost all cases. So, the knowledge of the value of v_{i-1} gives a lot of information about the probability over the values of v_i and the Bayesian measure assings a high quality to network structures containing these parent sets. Alternatively, v_{i+1} may be in π_i: when $v_{i+1} = 0$, we find in the database that $v_i = 0$ and when $v_{i+1} = 1$, we find that $v_i = 1$ for almost all cases. The only variable that would provide more information in combination with v_{i-1} or v_{i+1} about the value of v_i is w. However, adding w to the parent set of v_i would increase the number of different configuration of π_i considerably while the information obtained about the value of v_i would only increase for four cases in the database. Knowing only w would not give a significant amount of information about the value of v_i so π_i does not consist of w alone. Therefore, w will not be in v_i's parent set. For v_1 and v_n similar arguments hold. Since a network structure is a directed acyclic graph, there will be either an arc from v_i to v_{i+1}, $i = 1, \ldots, n \Leftrightarrow 1$, or the other way around.

We now turn to node w. We have that for w the parent set $\{v_{n-k+1}, \ldots, v_n\}$ gets assigned a higher quality according to the Bayesian measure than any other parent set of size k (see appendix Lemma 4.7 and 4.8). Furthermore, we have that if $\pi_w = \{v_k, \ldots, v_n\}$, for some $1 < k \leq n$, then B_S cannot be the network structure with highest quality for D_n since the structure with $\pi_w = \{v_{k-1}, \ldots, v_n\}$ for π_w has a better quality according to the Bayesian measure (see appendix Lemma 4.9). As a consequence, $\pi_w = \{v_1, \ldots, v_n\}$ results in the highest quality according to the Bayesian measure (see appendix Lemma 4.10). We conclude that for a database with N cases, a network structure with a parent set comprising $N/2$ variables can be assigned highest quality by the Bayesian measure.

So, while the asymptotic behavior of the Bayesian measure, information criteria, and MDL measure on databases of infinite size is the same, this is not the true for practical cases where only a finite-size database is available; the MDL measure and most information criteria will not assign the highest quality to network structures with parent sets of size larger than $\log N$ while the Bayesian measure may assign the highest quality to a network structure with a parent set of size $N/2$.

4.3 Search Strategies

One of the purposes of an algorithm for learning a network structure from data is to select a network structure with high quality according to a given quality measure. To this aim, one searches in the space of network structures for such a network structures. In this section, we will first introduce some terminology. First, we investigate the complexity of selecting a network structure with some optimality properties in Section 4.3.1. In Section 4.3.2, we review the most popular heuristics, K2 and B. To conclude, we show in Section 4.3.3 how to apply general purpose optimization algorithms for learning a network structure from data.

The search for a network structures with a high quality may be looked upon as a combinatorial optimization problem. Combinatorial optimization theory is engaged with the problem of selecting a solution with some optimality properties from among a large set solutions. We introduce some terminology from the theory of combinatorial optimization, following [1].

4.8 **Definition** An *instance of a combinatorial optimization problem* is a pair (f, S) where S is a *solution space* and $f : S \to \mathbb{R}$ is a real-valued function over S, called the *cost function*. A *global optimal solution* of (f, S) is an element $s \in S$ such that $f(r) \leq f(s)$ for all $r \in S$. □

A global optimal solution is also called *global optimum*. An element of S is called a *solution*. Now, let V be a set of variables. Learning a network structure from a database D over V can be formulated as an instance of a combinatorial problem: the problem that we consider is $(Q, \mathcal{B}_S)$ where the solution space $\mathcal{B}_S$ is the set of all possible network structures over V and the cost function Q is a quality measure.

4.9 **Definition** Let (f, S) be an instance of a combinatorial optimization problem. Then a *neighborhood structure* on (f, S) is a mapping $\mathcal{N} : S \to \mathcal{P}(S)$ where $\mathcal{P}(S)$ is the power set of S. □

The neighborhood structure defines for each solution $s \in S$ a set $\mathcal{N}(s) \subseteq S$ of solutions that are in some sense 'close' to s. The set $\mathcal{N}(s)$ is called the *neighborhood* of solution s. Each $r \in \mathcal{N}(s)$, r is called a *neighboring solution* or simply *neighbor* of s. For our combinatorial problem $(Q, \mathcal{B}_S)$, a neighborhood of a network structure B_S usually consists of a set of network structures that differ in only one arc from B_S.

4.10 **Definition** Let (f, S) be an instance of a combinatorial optimization problem and let $\mathcal{N}$ be a neighborhood structure of (f, S). A *local optimum* of (f, S) is a solution $s \in S$ such that $f(s) \leq f(r)$ for all $r \in \mathcal{N}(s)$. □

n	$G(n)$
0	1
1	1
2	3
3	25
4	543
5	29.281
6	3.781.503
7	1.138.779.265
8	783.702.329.343
9	1.213.442.454.842.881
10	4.175.098.976.430.598.143

Figure 4.6: The number of directed acyclic graphs for small number of nodes.

A *generation mechanism* is a means of selecting a solution r from $\mathcal{N}(s)$. In general, search for a network structure with high quality amounts to iteratively applying a generation mechanism to a neighborhood structure. The network structures with the highest quality visited during this search is the one returned by the algorithm.

4.3.1 Search Complexity

As pointed out in the previous section, learning a network structure amounts to selecting a network structure with a relatively high quality from among the set of all possible network structures. The number of different network structures over n different nodes is given by the following recursive formula [91]:

$$G(n) = \begin{cases} 1 & \text{if } n = 0, \\ \sum_{i=1}^{n} (\Leftrightarrow 1)^{i+1} \binom{n}{i} 2^{i(n-1)} G(n \Leftrightarrow i) & \text{if } n > 0. \end{cases}$$

Table 4.6 shows the number $G(n)$ for some small values of n. As the number of different network structures grows exponentially in the number of nodes, it is evident that it is not feasible from a computational point of view to consider all network structures. The question arises whether or not an efficient algorithm exists for selecting an optimal network structure. We will show that the problem of selecting a network structure with certain minimality properties is NP-hard.

4.5 **Lemma** Let V be a set of n variables and let Pr be a known joint probability distribution over V. Let an oracle be available that reveals whether an independency statement holds in Pr or not. Let $k \leq |V|$ be a positive constant and let $s = \frac{1}{2}n(n \Leftrightarrow 1) \Leftrightarrow \frac{1}{2}k(k \Leftrightarrow 1)$. Then, the problem of deciding whether or not there is an I-map of Pr with less or equal to s arcs by consulting the oracle is NP-complete. □

Proof: Let OBUILD be the problem of deciding whether or not an I-map of Pr with less or equal than s arcs exists where an oracle is available that reveals whether an independency statement holds or not.

First, we show that OBUILD is in NP. To this end we show that a nondeterministic Turing machine can solve OBUILD in polynomial time. The machine needs only to guess a network structure B_S. Deciding whether B_S is an I-map of Pr by consulting an oracle can be performed by constructing a topological ordering on B_S and asking the oracle whether $I(v_i, \pi_i, V_i \backslash \pi_i)$ is valid for all $v_i \in V$. All these actions can be performed in polynomial time. Deciding whether B_S has less than s arcs can be performed in polynomial time as well. So, a nondeterministic machine can decide OBUILD in polynomial time, and we conclude that OBUILD is in NP.

Next, we reduce the independent set problem GT20, which is known to be NP-complete [41], to OBUILD and show that this reduction can be performed in polynomial time.

GT20 can be stated as follows. Let $G = (V, E)$ be an undirected graph and $k \leq |V|$ be a positive integer. GT20 is the problem of deciding whether G contain an independent set of size k or more, that is, whether a set $V' \subseteq V$ exists such that $|V'| \geq k$ and no two nodes in V' are adjacent.

The reduction of instances of GT20 to instances of OBUILD can be performed as follows. Let $G = (V, E)$ be an undirected graph; note that we do not interpret G as a Markov network. Now, let M^I be the independency model $\{I(X, Z, Y) | \forall_{u,v \in XYZ}(u, v) \notin E\}$; in the appendix we show that a probability distribution exists that induces such an independency model M^I. By inspection we find that M^I is a graphoid independency model.

An oracle that reveals whether $I(X, Z, Y)$ is in M^I can check for all $u, v \in XYZ$ whether (u, v) is an edge in G, which is a polynomial time task. So, the transformation of GT20 to OBUILD can be performed in polynomial time.

Let G be an undirected graph and consider the instance of GT20 for G and the instance of OBUILD by transforming G as described above. Assume an independent set of size at least k exists in G and let $S \subseteq V$, $k = |S|$, be an independent set of nodes in G, that is, a set of nodes such that for all $u, v \in S$ we have that $(u, v) \notin E$. Consider a causal input list $L_{<_V}$ over M^I.

If $<_V$ is such that the nodes in S are assigned the numbers 1 up to k, then all parent sets of these nodes would be empty, since the construction of M^I requires that $I(v_i, \emptyset, V_i) \in M^I$ if $\forall_{u,v \in V_i \cup \{v_i\}}(u, v) \notin E$. So, the network structure B_S corresponding to $L_{<_V}$ contains at most $\frac{1}{2}n(n \Leftrightarrow 1) \Leftrightarrow \frac{1}{2}k(k \Leftrightarrow 1)$ arcs. Therefore, we have that if the instance of GT20 is true, then the instance of OBUILD is true.

Assume a network structure B_S can be found by consulting the oracle about M^I such that B_S has less than $\frac{1}{2}n(n \Leftrightarrow 1) \Leftrightarrow \frac{1}{2}k(k \Leftrightarrow 1)$ arcs and B_S is an I-map of Pr. Let $<_V$ be a topological ordering on B_S. Now there exists a node v_i such that $|V_i| \geq k \Leftrightarrow 1$ where there is a node $v_j \in V_i$ such that $v_j \notin \pi_i$. If no such node exists, then B_S would have to have more than $\frac{1}{2}n(n \Leftrightarrow 1) \Leftrightarrow \frac{1}{2}k(k \Leftrightarrow 1)$ arcs. And since $\langle v_i, V_i \backslash v_j, v_j \rangle$ in B_S and B_S is an I-map of Pr we have that $I(v_i, V_i \backslash v_j, v_j)$ is in M^I. By construction of M^I, we then have that for all $u, v \in v_i V_i$ $(u, v) \notin G$. Since $|v_i V_i| \geq k$, $v_i V_i$ is an independent set in G. Therefore, we have that if the instance of OBUILD is true, then instance of GT20 is true. □

As a result of Lemma 4.5 we have that selecting an I-map of a distribution Pr with a minimal number of arcs by consulting an oracle that reveals conditional independence information about Pr is expected to have a computational complexity that is exponential in the number of variables in V. Furthermore, we have the following property.

4.7 **Theorem** Let V be a set of n binary variables and let Pr be a known joint probability distribution over V. Let an oracle be available that reveals whether an independency statement holds in Pr or not. Let k be a positive constant and let $s = k + 2^n \Leftrightarrow 2^k$. Then, the problem of deciding whether or not there is an I-map which has at most s associated parameters by consulting the oracle is NP-complete. □

Proof: Let NBUILD be the problem of deciding whether or not an I-map of Pr which has at most s associated parameters exists, where an oracle is available that reveals whether an independency statement holds or not.

Note that a network structure in which k nodes have empty parent-sets has associated at most $k + \sum_{i=k+1}^{n} 2^{i-1}$ parameters. Now, $k + \sum_{i=k+1}^{n} 2^{i-1} = k + \sum_{i=1}^{n} 2^{i-1} \Leftrightarrow \sum_{i=1}^{k} 2^{i-1}$ which can be written as $k + 2^n \Leftrightarrow 2^k$. Therefore, a solution of the OBUILD problem presented in the proof of Lemma 4.5 for certain k is equivalent to a solution of the NBUILD problem for the same k. □

As a result of Lemma 4.5 we have that selecting a minimum I-map of a distribution Pr over binary variables by consulting an oracle that reveals conditional independence information about Pr is expected to have a computational complexity that is exponential in the number of variables in V. We conjecture that a similar result holds in the case that V contains non-binary variables. From Lemma 4.5 and Theorem 4.7 we have that no polynomial time algorithm may be expected to exist for finding an optimal network structure exhibiting certain minimallity properties, not even when the distribution to be represented is known and conditional independencies can be determined with certainty. Since from data conditional independencies cannot be deduced reliably, learning network structures from data is an even harder task to fulfill. Therefore, we do not expect to find polynomial time algorithms for selecting network structures with highest quality. In fact, finding such a network structure has been proved to be NP-hard [52].

4.3.2 Search Heuristics

Considering the results in the previous subsection, it is apparent that search heuristics are necessary for solving the problem of selecting a network structure with high quality. Several heuristics have been proposed in literature [14, 23]. We will present these heuristics and some generalizations in this subsection.

The heuristics gain their computational efficiency by only partially calculating the quality for each network structure considered. Close examination of the various quality measures namely reveals that a network structure's quality with respect to a given database can be expressed as the sum of the of the contributions of the separate nodes and some constant C.

4.11 **Definition** Let V be a set of variables. Let B_S be a network structure over V and let D be a database over V. Let v_i, r_i, π_i, q_i, N_{ij} and N_{ijk} be as before.

The *Bayesian quality of node* v_i in B_S and D is

4.21
$$m^B(v_i, \pi_i, D) = \sum_{j=1}^{q_i} \left(\log \frac{(r_i \Leftrightarrow 1)!}{(N_{ij} + r_i \Leftrightarrow 1)!} + \sum_{k=1}^{r_i} \log N_{ijk}! \right).$$

The *information criterion quality of node* v_i in B_S and D is

4.22
$$m^{IC}(v_i, \pi_i, D) = \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \cdot \log \frac{N_{ijk}}{N_{ij}} \Leftrightarrow q_i \cdot (r_i \Leftrightarrow 1) \cdot f(N).$$

The *MDL quality of node* v_i in B_S and D is

4.23
$$m^{MDL}(v_i, \pi_i, D) = \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \cdot \log \frac{N_{ijk}}{N_{ij}} \Leftrightarrow \frac{1}{2} q_i \cdot (r_i \Leftrightarrow 1) \cdot \log N.$$

□

When the context makes clear which database D is considered, we omit the parameter D in the notation of the quality of a node and write $m^B(v_i, \pi_i)$, $m^{IC}(v_i, \pi_i)$, and $m^{MDL}(v_i, \pi_i)$ to denote $m^B(v_i, \pi_i, D)$, $m^{IC}(v_i, \pi_i, D)$, and $m^{MDL}(v_i, \pi_i, D)$ respectively. When we speak of the *quality of node* v_i in general, irrespective of the quality measure employed, we will write $m(v_i, \pi_i, D)$ or $m(v_i, \pi_i)$ for short.

4.6 **Lemma** Let V be a set of variables, and let D be a database over V. Let Q be either the Bayesian measure, the MDL measure, or an information criterion. When the prior on all network structures is uniform, then $Q(B_S, D)$ can be written as

$$C + \sum_{i=1}^{n} m(v_i, \pi_i, D)$$

for any network structure B_S over V. □

Proof: We show that $B(B_S, D)$ can be written as $C + \sum_{i=1}^n m^B(v_i, \pi_i, D)$. The proofs for the MDL measure and information criterion are analogues. By definition, $B(B_S, D)$ is $\log P(B_S, D)$ for any network structure B_S over V, which can be written as $\log \left(P(B_S) \cdot \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \cdot \prod_{k=1}^{r_i} N_{ijk}! \right)$. This formula can be reformulated to $\log P(B_S) + \sum_{i=1}^n \sum_{j=1}^{q_i} \left(\log \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} + \sum_{k=1}^{r_i} \log N_{ijk}! \right)$. Now, since the prior on all network structures is uniform, $\log P(B_S)$ can be replaced by a constant C. Furthermore, by definition of the Bayesian quality of node we have that $Q(B_S, D)$ can be written as $C + \sum_{i=1}^n m(v_i, \pi_i, D)$. □

The property in Lemma 4.6 will be referred to as the *sum property*. The sum property is very helpful for comparing two network structures B_S and $B_{S'}$ that differ only slightly. For example, suppose that B_S and $B_{S'}$ differ only in the sense that $B_{S'}$ contains an extra arc $v_j \rightarrow v_i$. Now, let π_i and π_i' denote the parent sets of node v_i in B_S and $B_{S'}$, respectively. Then, $Q(B_S, D) \Leftrightarrow Q(B_{S'}, D) = C + \sum_i m(v_i, \pi_i) \Leftrightarrow C \Leftrightarrow \sum_i m(v_i, \pi_i') = m(v_i, \pi_i) \Leftrightarrow m(v_i, \pi_i')$. So, with very little computational effort, B_S and $B_{S'}$ can be compared with respect to their quality. In the rest of this section, we assume that the prior distribution over all network structures $P(B_S)$ is uniform and thus that all considered quality measures have the sum property.

```
Algorithm K2 ($<_V$)

for $i = 1, \ldots, n$ do                {initialize}
    $\pi_i \leftarrow \emptyset$
for $i = 2, \ldots, n$ do                {main loop}
    repeat
        select $v \in \{v_1, \ldots, v_{i-1}\} \backslash \pi_i$ that
            maximizes $g = m(v_i, \pi_i \cup \{v\})$
        $\Delta \leftarrow g \Leftrightarrow m(v_i, \pi_i)$
        if $\Delta > 0$ then
            $\pi_i \leftarrow \pi_i \cup \{v\}$
    until $\Delta \leq 0$ or $\pi_i = \{v_1, \ldots, v_{i-1}\}$
return $\pi_1, \ldots, \pi_n$
```

Greedy heuristics K2

Based on the Bayesian measure, Cooper and Herskovits have developed a search heuristic for learning network structures from data, called *K2* [23]. This algorithm, however, can also be used with other quality measures. The K2 algorithm takes an ordering on the variables involved and performs a greedy search in the sense that the arc that obeys the ordering and maximally increases the quality is added, until no quality increase is possible by adding a single arc.

Let V be a set of variables. Consider the instance of a combinatorial problem $(Q, \mathcal{B}_S)$ where $\mathcal{B}_S$ is the set of all possible network structures over V. Let $<_V$ be an ordering on V. The neighborhood of a network structure $B_S = (V, E)$ is the set of all network structures that can be obtained from B_S by adding a single arc $v_j \rightarrow v_i$ to B_S such that $v_j <_V v_i$, that is,

$$\mathcal{N}(B_S) = \{(V, E') | E' = E \cup (v_j, v_i), (v_j, v_i) \notin E, v_j <_V v_i\}.$$

The algorithm starts with the arc-less network structure $B_S = (V, \emptyset)$. Given a network structure B'_S, the generation mechanism iteratively selects a network structure $B_{S''} \in \mathcal{N}(B'_S)$ such that $Q(B_{S''}, D) \geq Q(B_{S'''}, D)$ for all $B_{S'''} \in \mathcal{N}(B'_S)$. The iteration is stopped if a local optimum is reached.

The generation mechanism can be implemented efficiently. Due to the sum property, all nodes can be considered independent of each other. Further, instead of complete network structures, only a node with its parents need to be considered. For each node $v_i \in V$, a parent set is calculated by starting with the empty parent set and successively adding to the parent set a node v_j and that is lower ordered than v_i and maximally improves the quality of v_i. This process is repeated until adding such nodes does not increase the quality of node v_i anymore or the parent set consists of all lower ordered variables. Below the K2 heuristic is detailed in pseudo-code. Note that node v_1 need not be considered, because $\pi_1 = \emptyset$ for any network structure.

The heuristic K2 will not always return a (minimal) I-map, not even for large databases where the behavior of a quality measure can be interpreted as an conditional-independency test. For example, let $V = \{v_1, v_2, v_3\}$ be a set of variables and let $<_V$ be some ordering on V. Furthermore, let Pr be a joint probability distribution on V such that all variables are pairwise independent but dependent given the third variable. Now consider a large enough database generated with Pr. K2 will not return a minimal I-map but an empty graph instead. Note that this implies that K2 will not necessary return a network structure with the highest quality that obeys the given ordering. However, if a P-map exists for the underlying distribution, K2 will return an I-map.

A major drawback of K2 is that it relies on an ordering on the nodes which influences the resulting network structure and its quality to a large extent. To guarantee a good performance with K2, it is essential to choose a 'good' ordering. Such an ordering, for example, may be provided by a domain expert. Recall, however, that the main aims of learning belief networks from data are to shorten the build-test cycle, and to circumvent a lengthy knowledge acquisition process. Alternatively, K2 can be applied with a randomly chosen ordering after which depending on the returned network structures this ordering is optimized in a post-processing step. We refer the reader to [7] for further details.

Algorithm B

Buntine has proposed a search heuristic that does not require an ordering on the variables involved [14]; we call this heuristic *algorithm B*. Like K2, algorithm B is a greedy search heuristic that exploits the sum property of the various quality measures. The main difference between K2 and B is that in B no ordering on the variables is required. The neighborhood of a network structure B_S consists of all graphs that have one extra arc with respect to B_S such that no cycle occurs in the graph.

Let V be a set of variables. Consider the instance of a combinatorial problem $(Q, \mathcal{B}_S)$ where $\mathcal{B}_S$ is the set of all possible network structures over V. The neighborhood of a network structure $B_S = (V, E)$ is the set of network structures that can be obtained from B_S by adding a single arc (v_i, v_j) to B_S such that the new network structure is a directed acyclic graph, that is, $\mathcal{N}(B_S)$ is equal to

$$\{(V, E') | E' = E \cup (v_j, v_i), (v_j, v_i) \notin E, (V, E') \text{ is a directed acyclic graph}\}.$$

Like K2, the algorithm starts with the arc-less network structure $B_S = (V, \emptyset)$. Given a network structure $B_{S'}$, the generation mechanism again selects a network structure $B_{S''} \in \mathcal{N}(B_{S'})$ such that $Q(B_{S''}, D) \geq Q(B_{S'''}, D)$ for all $B_{S'''} \in \mathcal{N}(B'_S)$. The algorithm is stopped if a local optimum is reached.

We take a closer look at the generation mechanism. Let B_S be the network structure at some moment during the execution of algorithm B. To find a network structure from the neighborhood of B_S with highest quality, the generation mechanism needs to determine an arc that upon addition to B_S gives the highest increase in quality and does not introduce a cycle. Now, let B_{Sij} denote the network structure obtained by adding the arc $v_j \rightarrow v_i$ to B_S that does not introduce a cycle. From the sum property of quality measures, we have that the difference in quality between B_S and B_{Sij} equals $Q(B_S, D) \Leftrightarrow Q(B_{Sij}, D) = m(v_i, \pi_i) \Leftrightarrow m(v_i, \pi_i v_j)$ where π_i is the parent set of v_i in B_S. In the pseudo-code of algorithm B shown below, these values

```
**Algorithm B**

for $i \in \{1,\ldots,n\}$ do  $\pi_i = \emptyset$        {initialize}
for $i = 1,\ldots,n$, $j = 1,\ldots,n$ do
    if  $i \neq j$ then
        $A[i,j] \leftarrow m(v_i,\{v_j\}) \Leftrightarrow m(v_i,\emptyset)$
    else
        $A[i,j] \leftarrow \Leftrightarrow\infty$    {obstruct $v_i \rightarrow v_i$}

repeat                                {main loop}
    select $i$, $j$ that maximize $A[i,j]$
    if $A[i,j] > 0$ then
        $\pi_i \leftarrow \pi_i \cup \{v_j\}$
        for $a \in A_i$, $b \in D_i$ do
            $A[a,b] \leftarrow \Leftrightarrow\infty$   {obstruct introduction of cycles}
        for $k \leftarrow 1$ to $n$ do
            if $A[i,k] > \Leftrightarrow\infty$ then
                $A[i,k] \leftarrow m(v_i,\pi_i \cup \{v_k\}) \Leftrightarrow m(v_i,\pi_i)$
until $A[i,j] \leq 0$ or $A[i,j] = \Leftrightarrow\infty$ for all $i$, $j$
return $\pi_1,\ldots,\pi_n$
```

are stored in the array A; in this array we have $A[i,j] = m(v_i,\pi_i v_j) \Leftrightarrow m(v_i,\pi_i)$ if addition of $v_j \rightarrow v_i$ does not introduce a cycle and $A[i,j] = \Leftrightarrow\infty$ otherwise. Note that if subsequently an arc $v_m \rightarrow v_k$ is added, only the values $A[k,m]$, $m = 1,\ldots,n$, need to be recalculated. Also note that if at some moment adding an arc $v_m \rightarrow v_k$ to the network structure would introduce a cycle, then it will introduce a cycle at any moment later on. So, after $A[k,m]$ is set to $\Leftrightarrow\infty$ it never is changed. In the pseudo-code, A_i denotes the set of indices of the ascendants of v_i and D_i denotes the set of indices of descendants of v_i including i. The largest computational effort is necessary for calculating the values for the array A. This calculation is performed every time an arc is added for at most n nodes. Maximally $\frac{1}{2}n(n \Leftrightarrow 1)$ are added. So, the computational complexity of algorithm B in terms of calculation of m_i is $O(n^3)$.

Just like K2, algorithm B does not always return a minimal I-map. For example, let $V = \{v_1,v_2,v_3\}$ be a set of variables and let $<_V$ be some ordering on V. Furthermore, let Pr be a joint probability distribution on V such that all variables are pairwise independent but dependent given the third variable. Now consider a large enough database generated with Pr. Algorithm B will not return a minimal I-map but an empty graph instead. Note that this implies that algorithm B will not necessary return a network structure that obeys the given ordering with the highest quality. So, it is advised to optimize the ordering of the variables as implied by the structure generated by algorithm B in a post-processing.

K2 and B with k Step Look-Ahead

The heuristics K2 and B may be looked upon as search algorithms that look only one step ahead in the search process: a single arc is selected that maximizes the quality increase irrespective of a quality increases obtained by selecting larger sets of arcs. As argued in the preceding section, these heuristics will not always return a minimal I-map, not even if large databases are available. The reason for this behavior is that these heuristics do not consider a quality increase obtained by selecting sets of arcs larger than one. These algorithms can be extended to adopt a k-step look-ahead. The basic idea of this extension is to select a group of k or less arcs, instead of a single arc, that increases the quality most.

Again, the algorithms starts with the arc-less network structure. Given a network structure B_S, the generation mechanism selects the network structure $B_{S'} \in \mathcal{N}(B_S)$ that maximizes $Q(D, B_{S'})$. The difference with the heuristics K2 and B is that another neighborhood structure is used.

For *K2 with k step look-ahead*, the neighborhood of a network structure $B_S = (V, E)$ is defined as follows, $\mathcal{N}(B_S)$ is

$$\{(V, E') | E' = E \cup F, F = \{(v_j, v_i) | v_j, v_i \in V(v_j, v_i) \notin E, v_j <_V v_i\}, 1 \leq |F| \leq k\}.$$

As the introduction of nodes in the parent set of a node does not influence the selection of parent sets for all other nodes, it suffices to consider the addition of a group of k or less arcs ending in a single node v_i, that is, $\mathcal{N}(B_S)$ is

$$\{(V, E') | E' = E \cup F, v_i \in V, F = \{(v_j, v_i) | v_j \in V, (v_j, v_i) \notin E, v_j <_V v_i\}, 1 \leq |F| \leq k\}.$$

For *B with k step look-ahead* the neighborhood of a network structure $B_S = (V, E)$ is

$$\mathcal{N}(B_S) = \{(V, E') | E' = E \cup F, F = \{(v_i, v_j) | (v_i, v_j) \notin E\}, 1 \leq |F| \leq k,$$

$$(V, E') \text{ is a directed acyclic graph}\}.$$

Now observe that, unlike for K2, the introduction of nodes in the parent set of a node v_i influences the selection of parent sets for some other nodes; if node v_j is selected for inclusion in π_i, then v_i cannot be selected for π_j. However, to construct $\mathcal{N}(B_S)$ for $k = 2$ and for the initial empty network structure B_S, we can select a single arc in $n \cdot (n \Leftrightarrow 1)$ ways, and two arcs that do not form a cycle in $n \cdot (n \Leftrightarrow 1) \cdot n \cdot (n \Leftrightarrow 2)$ ways. So, $\mathcal{N}(B_S)$ would be of size $O(n^4)$, and in general $O(n^{2k})$. As a result, the computational complexity of an algorithm that uses this neighborhood structure would become very large for larger k. Therefore, we consider the addition of groups of arcs that end in the same node. More formally, we use the following neighborhood for a network structure $B_S = (V, E)$;

$$\mathcal{N}(B_S) = \{(V, E') | E' = E \cup F, v_i \in V, F = \{(v_i, v_j) | v_j \in V, (v_j, v_i) \notin E\}, 1 \leq |F| \leq k,$$

$$(V, E') \text{ is a directed acyclic graph}\}.$$

Note that the heuristics with k step look-ahead for $k > 1$ are computational less attractive than ordinary K2 and B. On the other hand, the search space is explored more effectively. We will try to find experimentally a suitable choice of k such that these effects are balanced. The heuristics with k step look-ahead will not necessarily return a minimal I-map, not even when very large databases are available. As a result, these heuristics will not always return a network structure with highest quality.

4.3.3 General Approaches

In the previous subsection, we have reviewed several well-known search heuristics that have been designed especially for the purpose of learning network structures from data. A major drawback of these heuristics is that the network structure returned not necessarily has maximal quality by the quality measure that is used.

In this subsection, we consider application of the general purpose optimization algorithms tabu search, simulated annealing, and rejectionfree annealing for learning network structures from data. Unlike the heuristics, these algorithms have the any time property, that is, these algorithms can be stopped at any desired moment while it may be expected that the longer the allowed execution time, the better the returned result. Furthermore, theoretical results concerning global optima are available. Genetic programming, another popular optimization method, is not considered because the method is sensitive to many parameters, though its use have been reported [18, 70].

Neighborhood Structures for General Approaches

A basic condition for general search algorithms to be applicable to a combinatorial optimization problem (f, S) is that the cost function f can be easily calculated for elements of the neighborhood of a solution. So, an appropriate neighborhood structure that fulfills this condition should be chosen. First of all, we do not demand an ordering on the nodes, unlike we did for K2. Without ordering, a global optimum may be selected.

For the greedy heuristics discussed in the previous section, the neighborhood structure defined for a network structure B_S, network structures that are obtained from B_S by adding a single arc that does not introduce cycles in the new network structure seemed to be appropriate. However this neighborhood structure does not allow for removing arcs, and one cannot escape from a local optimum. For the general-purpose optimization algorithms, we therefore build on another neighborhood structure where the neighborhood of a network structure B_S contains not only all structures obtained by adding one arc to B_S but also all network structures that can be obtained from B_S by removing a single arc. So, if $B_S = (V, E)$, then $\mathcal{N}(B_S)$ is defined as

$$\{(V, E')|E' = E \cup (v_j, v_i),\ (v_j, v_i) \notin E, (V, E') \text{ is a directed acyclic graph}\}$$

$$\cup\ \{(V, E')|E' = E \backslash (v_j, v_i), (v_j, v_i) \in E\}.$$

We will refer to this neighborhood as the *straight neighborhood* of B_S. Note that due to the sum property of the quality measures considered, the difference in quality of B_S and a network structure $B_{S'} \in \mathcal{N}(B_S)$ can be computed efficiently since B_S and

$B_{S'}$ have different parent sets for one node only, say v_i: if π_i and π'_i are the parent sets of v_i in B_S and $B_{S'}$, respectively, then $Q(B_S, D) \Leftrightarrow Q(B_{S'}, D) = m(v_i, \pi_i) \Leftrightarrow m(v_i, \pi'_i)$.

Consider algorithm B, when $v_j \rightarrow v_i$ is the first arc that is added, and the quality measure is an information criterion. Then, the increase in quality is equal to when $v_i \rightarrow v_j$ is added so no distinction in the quality is made for the direction of the arc. It may very well be that the direction of the arc between nodes v_i and v_j is chosen erroneously and need to be reversed. Using the straight neighborhood structure, arc reversal can be reached in two steps: first the arc $v_i \rightarrow v_j$ is removed, and then the reversed arc $v_j \rightarrow v_i$ is added. As the intermediate network structure may have a much lower quality than the original one it may take a very long time before the $v_i \rightarrow v_j$ is removed. To circumvent this problem, we extend the neighborhood of a network structure B_S with all network structures that can be obtained from B_S by reversing a single arc. So, if $B_S = (V, E)$, then $\mathcal{N}(B_S)$ is defined as the straight neighborhood of B_S united with

$$\{(V, E')|E' = E \cup (v_j, v_i)\backslash(v_i, v_j), (v_i, v_j) \in E\}.$$

We call this the *reversed neighborhood* of B_S. Again, comparison of the quality of B_S and a network structure $B_{S'} \in \mathcal{N}(B_S)$ can be performed efficiently. If $B_{S'}$ is also in the straight neighborhood structure of B_S, then we have that $Q(B_S, D) \Leftrightarrow Q(B_{S'}, D) = m(v_i, \pi_i) \Leftrightarrow m(v_i, \pi'_i)$, as before. Otherwise, B_S and $B_{S'}$ have different parent sets for two nodes only, say v_i and v_j; now if π_i and π'_i are the parent sets of v_i in B_S and $B_{S'}$, respectively, and π_j and π'_j have the same meaning for v_j, then, $Q(B_S, D) \Leftrightarrow Q(B_{S'}, D) = m(v_i, \pi_i) \Leftrightarrow m(v_i, \pi'_i) + m(v_j, \pi_j) \Leftrightarrow m(v_j, \pi'_j)$.

The straight and reversed neighborhood of a network structure B_S contain network structures that differ in just one arc. Similar to the extension of the neighborhood structure for the heuristics with k step look-ahead, we could extend the straight and reversed neighborhood structure. For tabu search and rejectionfree annealing, in each iteration the complete neighborhood of a current network structure is considered. Since extended neighborhoods would become very large, we will not consider such an extension to keep the computational complexity of the algorithms feasible.

We will discuss the various general purpose search algorithms for the straight neighborhood structure only. However, they are straightforward generalized for the case the reversed neighborhood structure is used.

Tabu Search

Tabu search is a general-purpose combinatorial optimization algorithm [46]. The basic idea is to start with an arbitrary solution and recursively select a new solution from the neighborhood of the previous one that maximally increases a pre-defined cost function. If there is no solution that increases the cost function, the present solution is a local optimum. The search process leaves this local optimum by selecting a solution that minimally decreases the cost function. If a solution is selected that decreases the cost function, immediate re-selection of the local optimum just visited is prevented by maintaining a list of solutions that are forbidden, the so called *tabu-list*. The length *tll* of this tabu-list determines how many iterations it may take to return to a local optimum; the higher *tll*, the longer it will take to possibly return

```
Tabu Search $(tll, stopcriterion)$

initialize $s$
$s_{best} \leftarrow s$
$tp \leftarrow 0$
for $i \leftarrow 0$ to $tll \Leftrightarrow 1$ do $tabu\_list[i] \leftarrow s$

repeat
    select solution $r$ from $\mathcal{N}(s)$ that maximizes $f(r)$ and is not in $tabu\_list$
    $s \leftarrow r$
    $tabu\_list[tp] \leftarrow r$
    $tp \leftarrow (tp + 1)$ mod $tll$
    if $f(s) > f(s_{best})$ then $s_{best} \leftarrow s$
until stopcriterion
return $s_{best}$
```

to the local optimum, the larger the probability that it will not be visited again. The solution returned by tabu search is the best solution visited during execution of the algorithm. Tabu search is attractive for its simplicity, the few parameters that are necessary, and the empirical results which look very promising for a variety of combinatorial optimization problems [59, 111].

Below the general tabu-search algorithm is shown in pseudo-code. The tabu-list length tll and a stop-criterion depend on the instance of the combinatorial optimization problem (f, S). The tabu-list of length tll is usually implemented as a circular list; an array with tll elements and an index tp that points to the last element inserted.

For the search generally a stop criterion is employed. This criterion differs among various applications. For example, the search can be stopped after a fixed number of iterations [59], or when during the last fixed number of steps no improvement has been made [116], or a combination of both [111].

For application of tabu-search to the problem of learning network structures from data, we build on the straight neighborhood structure defined above. The tabu-list employed identifies the last tll network structures. Note that as all network structures in the straight neighborhood $\mathcal{N}(B_S)$ of a network structure B_S differ only in a single arc, it suffices to store this arc and whether it has been added or removed.

Below the tabu-search algorithm adapted to selecting network structures is shown in pseudo-code. Note that we have once more used an array A to memorize changes in qualities. For representing the change in quality upon removal of an arc $v_j \to v_i$, we take $A[i,j] = m(v_i, \pi_i \backslash v_j) \Leftrightarrow m(v_i, \pi_i)$. Updating A after addition of an arc $v_j \to v_i$ is the same as before, except that for $v_k \in \pi_i$ in $A[i,k]$ the values $m(v_i, \pi_i \backslash v_k) \Leftrightarrow m(v_i, \pi_i)$ need to be entered. Note that when an arc $v_j \to v_i$ is deleted from a network structure, it is possible that the addition of an arc $v_m \to v_k$, that would introduce a cycle in the original structure, now becomes admissible. So, for all $A[k,m] = \Leftrightarrow\infty$, it is examined if addition of $v_m \to v_k$ is allowed, and if so, $A[k,m]$ is set to $m(v_k, \pi_k v_m) \Leftrightarrow m(v_k, \pi_k)$. The algorithm for updating A when the indices i and j

```
**Tabu Search** $(tll, stopcriterion)$

for $i = 1, \ldots, n$ do  $\pi_i \leftarrow \emptyset$ $\pi_{best,i} \leftarrow \emptyset$
$tp \leftarrow 0$
for $i = 0, \ldots, tll \Leftrightarrow 1$ do
    $tabu\_list[i] \leftarrow (0, 0)$ {enter dummy values}
for $i = 1, \ldots, n$, $j = 1, \ldots, n$ do
    if  $i \neq j$ then
        $A[i, j] \leftarrow m(v_i, v_j) \Leftrightarrow m(v_i, \emptyset)$
    else
        $A[i, j] \leftarrow \Leftrightarrow \infty$   {obstruct $v_i \rightarrow v_i$}

repeat
    select the indices $i, j$ that maximizes $A[i, j]$ and is not in $tabu\_list$
    if $v_j \in \pi_i$ then
        $\pi_i \leftarrow \pi_i \backslash v_j$
    else
        $\pi_i \leftarrow \pi_i v_j$
    $tabu\_list[tp] \leftarrow (i, j)$
    update_array($A$, $i$, $j$, $B_S$)
    $tp \leftarrow (tp + 1)$ mod $tll$
    if $\sum_{i=1}^{n} m(v_i, \pi_{best,i}) < \sum_{i=1}^{n} m(v_i, \pi_i)$ then $\forall_{i \in \{1,\ldots,n\}} \pi_{best,i} \leftarrow \pi_i$
until stopcriterion
return $\pi_{best,1}, \ldots, \pi_{best,n}$
```

have been selected and the current solution is network structure B_S is show below.

For finding the best network structure visited during execution of the algorithm, we have to test whether $\sum_{i=1}^{n} m(v_i, \pi_{best,i}) < \sum_{i=1}^{n} m(v_i, \pi_i)$. This test involves all nodes. One cannot simply return the parent sets that result in the best node quality because network structures containing cycles would arise.

Note that tabu search is started with an arc-less network structure just as the greedy heuristics K2 and B. If the straight neighborhood structure is used, tabu search at first will select the same network structures as in algorithm B. So, the network structure returned by tabu search is expected to have at least the quality of the network structure returned by algorithm B.

Simulated Annealing

Simulated annealing is a general purpose combinatorial optimization algorithm that was independently introduced by Kirkpatrick, Gelatt, and Vecchi [64] and by Černy [113]. The algorithm has been motivated by the process of annealing of metal to harden it. Annealing begins by heating the metal until it melts and all particles move freely. Then, the temperature is decreased carefully to allow the particles to arrange themselves in a highly structured lattice. Contrasting tabu search, a deep theoretical understanding of the mechanism underlying simulated annealing

```
update_array($A$, $i$, $j$, $B_S$)

for $k = 1, \ldots, n$ do
    if $v_k \in \pi_i$ then
        if $v_k \notin D_i$ then
            $A[i,k] \leftarrow m(v_i, \pi_i v_k) \Leftrightarrow m(v_i, \pi_i)$
        else
            $A[i,k] \leftarrow \Leftrightarrow\infty$
    else
        $A[i,k] \leftarrow m(v_i, \pi_i \backslash v_k) \Leftrightarrow m(v_i, \pi_i)$

if $v_j \notin \pi_i$ then
    for $v_k \in D_i$, $v_l \in A_i$ do
        $A[l,k] \leftarrow \Leftrightarrow\infty$
else
    for $k = 1, \ldots, n$ $l = 1, \ldots, n$ do
        if $A[k,l] == \Leftrightarrow\infty$ and $v_l \notin v_k$
        and adding $v_l \to v_k$ does not introduce a cycle then
            $A[i,k] \leftarrow m(v_i, \pi_i v_k) \Leftrightarrow m(v_i, \pi_i)$
```

has been developed [1, 77].

The basic idea of simulated annealing for solving an instance of an optimization problem (f, S) now is to assign to the problem a temperature T and interpret the cost $f(s)$ of a solution $s \in S$ as an energy level; such a solution s corresponds with the state of the metal. The algorithm now starts with an arbitrary solution and recursively selects a new solution r from the neighborhood $\mathcal{N}(s)$ of the current solution s. If the 'energy level' $f(r)$ is less than or equal to the current 'energy level' $f(s)$, then the new solution is accepted, that is, s is replaced by r. Otherwise, the new solution is accepted with probability $\exp \frac{f(r)-f(s)}{T}$. Note that the higher the temperature, the larger the probability that a new solution r will be accepted. The temperature varies during execution of the algorithm according to a cooling schedule. Initially, the temperature of the problem is so high that almost all solutions selected by the generation mechanism are accepted. During execution of the algorithm, temperature is decreased slowly so as to accept fewer and fewer solutions that are worse than the current solution. The algorithm returns the best solution found during execution of the algorithm. Below the simulated annealing algorithm is shown in pseudo-code. Note that if $f(r) > f(s)$ then $\exp(\frac{f(r)-f(s)}{T}) > \exp(0) > 1$, so the condition for accepting r will always be true.

The main parameter is the cooling schedule which consists of an initial and a final temperature, and a function describing the decrease in temperature over time. The final temperature can be looked upon as a stop-criterion for execution of the algorithm. Various cooling schedules an generation mechanisms have been proposed [1, 45, 60, 110]. However, selecting a satisfactory schedule in general is a difficult task; in fact, a lot of research effort has already been invested in this task.

For the cooling schedule we may adopt for example an empirical approach based

```
**Simulated Annealing** $(T_0, T_{end}, calc\_temp)$

initialize $s$
$T \leftarrow T_0$
$k \leftarrow 0$
$s_{best} \leftarrow s$

**repeat**
    select a solution $r$ from the solution space $S$
    **if** $\exp(\frac{f(s)-f(r)}{T}) > random[0..1]$ **then**
        $s \leftarrow r$
        **if** $f(s) > f(s_{best})$ **then** $s_{best} \leftarrow s$
    $k \leftarrow k + 1$
    $T \leftarrow$ *calc_temp*( $T$, $k$)
**until** $T < T_{end}$
return $s_{best}$
```

on the polynomial cooling schedule proposed by Aarts [1]. The initial temperature T_0 of the schedule is chosen in such a way that half of all selected solutions are accepted. This temperature is determined by sampling the solution space: a sample is selected and the average energy level is taken for the initial temperature. Note that since the quality measures grow linearly with the size of the database, it suffices to find an initial factor τ_0: given a database of size N, the initial temperature is set to $T_0 = \tau_0 \cdot N$. For the final temperature, we choose a value such that none of the solutions that are worse are accepted. We let the user define a number of iterations ni to make a comparison with the other algorithms in run-time and performance possible. So, we have $T_{ni} = \alpha^{ni} T_0$ and thus $\alpha = (T_{ni}/T_0)^{1/ni}$. Now, let T_k be the temperature at some artificial time k then T_k can be computed as $\alpha \cdot T_{k-1}$.

We start with a network structure without arcs. Then, we select a new network structure randomly from the straight neighborhood of the current network structure.[2] Following these choices, we arrive at the simulated annealing algorithm for selecting network structures with the straight neighborhood structure depicted below. Note that for the calculation of the difference of the quality of the current network structure B_S and that of the selected network structure $B_{S'}$ we distinguish between the cases where $B_{S'}$ has one arc extra and where $B_{S'}$ has one arc missing compared to B_S.

Rejectionfree Annealing

Rejectionfree annealing [50] is a general-purpose combinatorial optimization algo-

[2] For proofs of convergence of simulated annealing to the optimal solution, it is necessary to define a positive distribution Pr over all solutions and select a solution using Pr. However, with rationally defined distributions, seldomly a solutions is chosen that is not in the straight neighborhood. Since following the original approach would alter the algorithm only slightly but would ask for a more theoretical explanation, we adopt the simplified approach.

```
Simulated Annealing $(\tau_0, T_{ni}, ni)$

for $i = 1, \ldots, n$ do  $\pi_i \leftarrow \emptyset$ $\pi_{best,i} \leftarrow \emptyset$
$T \leftarrow \tau_0 \cdot N$
$\alpha \leftarrow (T_{ni}/T_0)^{1/ni}$
$k \leftarrow 0$

repeat
    repeat
        select two indices $i, j$ randomly
    until $v_j \in \pi_i$ or adding $v_j$ to $\pi_i$ does not introduce a cycle
    if $v_j \in \pi_i$ then
        if $\exp(m(v_i, \pi_i \backslash v_j) \Leftrightarrow m(v_i, \pi_i)) > random[0..1]$ then
            $\pi_i \leftarrow \pi_i \backslash v_j$
    else
        if $\exp(m(v_i, \pi_i v_j) \Leftrightarrow m(v_i, \pi_i)) > random[0..1]$ then
            $\pi_i \leftarrow \pi_i v_j$
    if $\sum_{i=1}^{n} m(v_i, \pi_{best,i}) < \sum_{i=1}^{n} m(v_i, \pi_i)$ then $\forall_{i \in \{1,\ldots,n\}} \pi_{best,i} \leftarrow \pi_i$
    $T \leftarrow T * \alpha$
    $k \leftarrow k + 1$
until $k = ni$
return $\pi_{best,1}, \ldots, \pi_{best,n}$
```

rithm that can be considered as an optimization of simulated annealing. At the end of the execution of the simulated annealing algorithm, many solutions are considered and then rejected. The idea of rejectionfree annealing is to avoid selecting solutions that will be rejected. To this end for each solution r in the neighborhood structure $\mathcal{N}(s)$ of the current network structure s, the probability $h(r)$ of r being accepted is calculated according to the acceptance criterion of the simulated annealing algorithm for certain temperature. When we take for simplicity a temperature of 1, we have that if $f(r) > f(s)$, then the acceptance probability $h(r) = 1$ and otherwise $h(r) = \exp(f(r) \Leftrightarrow f(s))$. Let p be the sum of those probabilities, that is, $p = \sum_{r \in \mathcal{N}(s)} h(r)$. Then, a probability distribution over the neighborhood structure is constructed by taking

4.24
$$P(r) = \frac{h(r)}{p}$$

for every $r \in \mathcal{N}(s)$. The algorithms now chooses a solution r in $\mathcal{N}(s)$ with probability $P(r)$ given by Formula 4.24. Below the pseudo-code for rejectionfree annealing is given. Note that the only parameter to the algorithm is a stop-criterion.

It has been shown that rejectionfree annealing is equivalent to simulated annealing [50] as soon as the probability of acceptance of new solutions drops below a certain threshold value which is determined by the temperature. Obviously, rejectionfree annealing is more efficient than simulated annealing since it does not consider solutions that are rejected. So, when the algorithm runs for a sufficiently long time, the probability that the global optimum will be found goes to one. Another advantage

```
Rejectionfree Annealing ($stopcriterion$)

initialize $s$

repeat
    calculate $h(r)$ for all $r \in \mathcal{N}(s)$
    select a solution $r$ from $\mathcal{N}(s)$ with probability $\frac{h(r)}{p}$
    $s \leftarrow r$
until stopcriterion
```

is that no parameters need to be chosen to define a cooling schedule. However, extra computational effort is necessary to calculate the probability distribution over the neighborhood structure.

Application of rejectionfree annealing to selection of network structures with high quality is performed similar to the application of simulated annealing to this task. Below the rejectionfree annealing algorithm for selecting network structures with the straight neighborhood structure is shown. Note that we have used an array B to memorize values of the probability of accepting the addition or removal of an arc similar to the memoization of $Q(B_S, D) \Leftrightarrow Q(B_{S'}, D)$ in the tabu-search algorithm; updating of B is performed with the update_array algorithm where calculation of the values of B is adapted.

4.3.4 Complexity of the Various Algorithms

In this section, we address the computational complexity of the various search algorithms. Note that learning minimal I-maps is NP-hard, which justifies the use of search heuristics for this task. One might argue that the complexity of the various search algorithms is of no interest because these algorithms are likely to be applied only once. However, with the use of a Bayesian belief network as knowledge-based system, new cases become available by which the database may be extended. The extended database can be used to evaluate the quality of the network structure and alternative better network structures may be suggested by one of the learning algorithms. Therefore, it is important to get a global idea of the run-time of the various search algorithms. We consider their complexity in the following paragraphs.

From [24] we have that the worst-case computational time complexity of the Bayesian quality of a node (Formula 4.21) is $O(N \cdot u \cdot r)$, where N is the size of the database, u is an upper bound on the size of a parent set, and r is the maximum of r_i, $i = 1, \ldots, n$. If the Bayesian measure is used, it is recommended to have a user-defined upper bound forced because of the effects discussed in Section 4.2.7. Of course, introduction of such an upper bound would affect the code for the algorithm slightly. By examining the definitions, we observe that the complexity of the IC quality and MDL quality of a node is the same as for the Bayesian quality. So, the worst computational time complexity of the calculation of the quality of a node is $O(N \cdot u \cdot r)$.

We now express the computational complexity of the algorithms K2 and B in

```
**Rejectionfree Annealing** $(ni)$

**for** $i = 1, \ldots, n$ **do** $\pi_i \leftarrow \emptyset$, $\pi_{best,i} \leftarrow \emptyset$
**for** $i = 1, \ldots, n$, $j = 1, \ldots, n$ **do**
    **if** $i \neq j$ **then**
        $B[i,j] \leftarrow \min(\exp(m(v_i, v_j) \Leftrightarrow m(v_i, \emptyset)), 1)$
    **else**
        $B[i,j] \leftarrow 0$    {obstruct $v_i \rightarrow v_i$}
$k \leftarrow 0$

**repeat**
    $p \leftarrow \sum_{i=1}^{n} \sum_{j=1}^{n} B[i,j]$
    $r \leftarrow random[0..p]$
    Select $i, j$ be such that $\sum_{(i',j')<(i,j)} B[i',j'] \leq r < \sum_{(i',j')\leq(i,j)} B[i',j']$
    **if** $v_j \in \pi_i$ **then**
        $\pi_i \leftarrow \pi_i \backslash v_j$
    **else**
        $\pi_i \leftarrow \pi_i v_j$
    update_array $B$
    $k \leftarrow k + 1$
    **if** $\sum_{i=1}^{n} m(v_i, \pi_{best,i}) < \sum_{i=1}^{n} m(v_i, \pi_i)$ **then** $\forall_{i \in \{1,\ldots,n\}} \pi_{best,i} \leftarrow \pi_i$
**until** $k = ni$
return $\pi_{best,1}, \ldots, \pi_{best,n}$
```

terms of the number of computations of the quality of a node. Both algorithms will calculate the quality of a node at most $O(n^2 \cdot u)$ times where u is an upper bound on the number of parents of a variable. The complexity of K2 and B is $O(n^2 \cdot u^2 \cdot N \cdot r)$.

For tabu search and rejectionfree annealing, initially for each solution in the neighborhood structure the quality of a node is calculated. There are at most $n \cdot (n \Leftrightarrow 1)$ elements in any straight neighborhood structure and also at most $2n \cdot (n \Leftrightarrow 1)$ elements in any reversed neighborhood structure. Therefore, initially the quality of a node must be calculated at most $4n \cdot (n \Leftrightarrow 1)$ times. With each iteration of the algorithm, the array that memorizes the qualities of nodes need to be updated for at on average at most n elements of the straight and n elements of the reversed neighborhood structure. The other operations in these algorithms are negligible. So, let ni be the number of iterations, then the complexity in terms of calculations of the quality of a node of tabu search and rejectionfree annealing, is $O(n^2) + O(n \cdot ni)$. Since $ni \gg n$ almost always, we have a complexity of $O(n \cdot ni)$. Incorporating the complexity of the calculation of the quality of a node gives a complexity of $O(n \cdot ni \cdot N \cdot u \cdot r)$.

For simulated annealing, negligible computational effort is necessary for initialization. In the main loop, the quality of a node is calculated twice. The loop is executed ni times and since the other operations are negligible, the complexity is $O(ni)$. Incorporating the complexity of the calculation of the quality of a node gives a complexity of $O(ni \cdot N \cdot u \cdot r)$.

In general, the number of iterations ni is much larger than $n \cdot u$ in tabu search and rejectionfree annealing. So, the heuristics K2 and B are computationally more efficient then those algorithms. Simulated annealing may seem to have a better worst case computational time complexity than K2 and B. However, many iterations are necessary for simulated annealing to return high quality network structures. Further, the difference between average and worst case complexity is smaller for simulated annealing than for K2 and B. As a result, K2 and B will in general be computational more efficient than simulated annealing.

4.3.5 Miscellaneous considerations

In the subsection on greedy heuristics it was argued that the algorithms K2 and B return local optima. To get a better network structure it may be useful to have a post-processing algorithm. The general-purpose algorithms can be used for this purpose; during the initialization of these algorithms not the arc-less graph but the network structure returned by K2 or B may to be used and for tabu search and rejectionfree annealing different values need to be calculated for the array A.

So far, we assumed that the sum property holds for the quality measure in use. Now, assume that the prior distribution over all network structures is positive. If B_S and $B_{S'}$ are two network structures that are equal except that $B_{S'}$ has an arc $v_j \to v_i$ that is not in B_S. Then, it is sufficient for determining $Q(B_S, D) \Leftrightarrow Q(B_{S'}, D)$ to calculate $P(B_S) \Leftrightarrow P(B_{S'}) + m(v_i, \pi_i) \Leftrightarrow m(v_i, \pi_i')$. If $P(B_S)$ is not positive several problem arise. For example, if all network structures with less than a fixed number $p \geq 2$ of arcs have zero prior probability, then algorithm K2 will always return an arc-less graph; addition of a single arc to the arc-less network structure does not change the quality in this case. The same problem arises with algorithm B. So, it may be necessary to change the initialization of the various algorithms to be sure that the algorithm does not return a non-informative network structure.

4.4 Probability Estimation

One of the aims of learning a Bayesian belief network from data is to decrease the time of the build-test cycle of constructing a belief network for use in a knowledge-based system. Now recall that the task of constructing a Bayesian belief network is twofold: constructing the network structure and defining the set of assessment functions. In the previous sections, learning network structures was addressed. In this section, we will investigate learning a set of assessment functions for a given network structure. We show how an learning technique known as smoothing can be easily incorporated into search heuristics thus yielding better estimates at small computational cost. But we will introduce direct estimation and smoothing first.

Let V be a set of variables. Let D be a database over V and let B_S be a network structure over V. Let v_i, π_i, r_i, N_{ij} and N_{ijk} as before. In learning the set of assessment functions, B_P for B_S, several probabilities of the form $Pr(v_i = x_{ik}|\pi_i = x_{\pi_i j})$ have to be estimated. Let θ_{ijk} denote the probability $Pr(v_i = x_{ik}|\pi_i = x_{\pi_i j})$ in the assessment function for a variable v_i. Cooper and Herskovits [24] showed that, under the assumptions with which the Bayesian measure was derived, the expected value of θ_{ijk} given the database D and network structure B_S is

4.25 $$E[\theta_{ijk}|B_S, D] = \frac{N_{ijk}+1}{N_{ij}+r_i}.$$

The expected value of θ_{ijk} can be used as an estimate for $\gamma_{v_i}(v_i = x_{ik}|\pi_i = x_{\pi_i j})$. When the assessment functions are obtained by setting $\gamma_{v_i}(v_i = x_{ik}|\pi_i = x_{\pi_i j}) = \frac{N_{ijk}+1}{N_{ij}+r_i}$ for all $i = 1, \ldots, n$, $j = 1, \ldots, q_i$, $k = 1, \ldots, r_i$, we say that these functions have been obtained by *direct estimation*. In the derivation of this expectation it is assumed that no prior information of the values θ_{ijk} is available. If prior information is available, it can be incorporated in the computation of the expected value of θ_{ijk} by adding cases to the database that expresses this information [24].

Furthermore, Cooper and Herskovits showed that

$$E[P(X|Y)|B_S, D] = P_B(X|Y)$$

for all $X, Y \subseteq V$, where P_B denotes the joint probability distribution over V represented by the Bayesian belief network $B = (B_S, B_P)$ where the assessment functions have been obtained by direct estimation from D. The expected value of $P(X|Y)$ is dependent not only on the database but also on the selected network structure B_S. If the network structure is not well-chosen, these expected values of $P(X|Y)$ can deviate considerably from the 'true' values. To circumvent this problem, it would be better to use $E[P(X|Y)|D]$ instead, which is,

4.26 $$E[P(X|Y)|D] = \sum_{B_S \in \mathcal{B}_S} E[P(X|Y), B_S|D] = \sum_{B_S \in \mathcal{B}_S} E[P(X|Y)|B_S, D] \cdot P(B_S|D),$$

where $\mathcal{B}_S$ is the set of all network structures over V and $P(B_S|D)$ is the probability

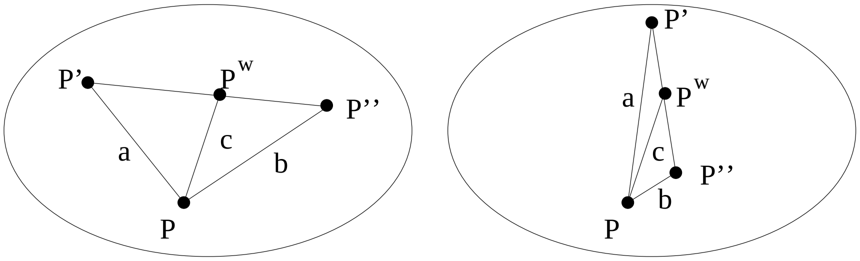


Figure 4.7: Relation between 'true' distribution P, estimated distributions P' and P'', and weighted distribution P^w.

of having network structure B_S given database D. This probability can be calculated using the Formula 4.1.

Since the cardinality of $\mathcal{B}_S$ is very large in general, it is not practical to consider all network structures in $\mathcal{B}_S$ for computing $E[P(X|Y)|D]$. This value, however, can be approximated by summing over network structures with high quality only. Even if only a small number of network structures is used, the deviation from the distribution P may be expected to be smaller than when a single network structure is used.

This effect is illustrated in Figure 4.7 which shows a part of the space of all possible distributions. Let P be the 'true' distribution, and let P' and P'' be two distributions represented by two different Bayesian belief networks B' and B'' respectively. The difference between the distributions P' and P is denoted by the vector $\vec{a}$, and the difference between P'' and P is denoted by $\vec{b}$. Now, let P^w be the distribution obtained by weighting the distributions P' and P'' based on Formula 4.26, and let $\vec{c}$ denote the difference between P^w and P. If $\vec{a}$ and $\vec{b}$ deviate from P in different dimensions as on the left-hand side of Figure 4.7, then $\vec{c}$ will have a smaller length than both $\vec{a}$ and $\vec{b}$, meaning that P^w is expected to deviate less from P than both P' and P''. If $\vec{a}$ and $\vec{b}$ point in the same direction from P as on the right-hand side of Figure 4.7, then the distance between P^w and P is still smaller than between P' and P. However, it is larger than between P'' and P. But, without extra information about P, it cannot be known which of P' and P'' is closest to P: the expected deviation of the distribution represented by B from the true distribution cannot be judged from the quality of the network structure and the database. So, it is safer to choose the weighted distribution P^w than to choose from P' and P'' involving of choosing the wrong one.

So, we can select a set of network structures, and estimate the assessment functions for each of these network structures. A distribution that is the weighted average over a collection of the thus obtained Bayesian belief networks can be expected to be closer to the real distribution than a distribution of a single Bayesian belief network. Because inference in just a single network is already NP-hard, it is computationally unattractive to have a set of belief networks instead of one. Under certain conditions, however, a collection of Bayesian belief networks and therewith a weighted distribution can be represented by a single Bayesian belief network. This technique is called *smoothing*.

Let V be a set of variables. Let S be a set of network structures over V and let

Π_i denote the set of parent sets π_i of node $v_i \in V$ that are found in the various network structures from S, that is, $\Pi_i = \{\pi_i | \pi_i$ is the parent set of v_i in $B_S, B_S \in S\}$. Furthermore, we demand an ordering $<_V$ on V exists that is obeyed by each network structure $B_S \in S$. Finally, we demand that S can be written as the set of all possible network structures that can be formed by selecting for each node $v_i \in V$ a parent set from Π_i. So, given for each $i = 1, \ldots, n$ the set Π_i, we can write $S = \{B_S | i \in \{1, \ldots, n\}, \pi_i \in \Pi_i\}$. Then, the weighted distribution Pr over V can be written as

$$Pr(V = x_V) = \alpha \sum_{B_S \in S} Pr_B(V = x_V | B_S, D) \cdot P(B_S|D),$$

where α is a normalizing constant. By using the definition of the Bayesian quality of a node, we find that $P(B_S, D) = P(B_S) \cdot 2^{\sum_{i=1}^{n} m^B(v_i, \pi_i)}$, where π_i is the parent set of v_i in B_S. Since $P(B_S|D) = P(B_D, S)/P(D)$ and $P(D)$ is a constant, we can write $P(B_D|S) = \beta P(B_S) \cdot 2^{\sum_{i=1}^{n} m^B(v_i, \pi_i)}$. Further, by expressing the joint probability distribution Pr_{B_S} in terms of the assessment functions and writing α' for $\alpha \cdot \beta$, we can write

$$Pr(V = x_V) = \alpha' \sum_{B_S \in S} \left(\prod_{i=1}^{n} \gamma_{v_i}(v_i = x_i | \pi_i = x_{\pi_i}) \cdot P(B_S) \cdot 2^{\sum_{i=1}^{n} m^B(v_i, \pi_i)} \right),$$

where $\gamma_{v_i}(v_i = x_i | \pi_i = x_{\pi_i})$ denotes the assessment function for node v_i in the Bayesian belief network with network structure B_S obtained by a direct estimate from D. Note that each value of the γ_{v_i} is the same for the same parent set of π_i and the same configuration of v_i; this value does not depend on the other parent sets. Note that this function is the same for all network structures where the parent set of v_i is the same. Since the distribution over all network structures is uniform, we have that $P(B_S)$ is a constant. By writing α'' for $\alpha' \cdot P(B_S)$ and by grouping terms, we find

$$Pr(V = x_V) = \alpha'' \cdot \sum_{B_S \in S} \left(\prod_{i=1}^{n} \gamma_{v_i}(v_i = x_i | \pi_i = x_{\pi_i}) \cdot 2^{m^B(v_i, \pi_i)} \right).$$

By an inductive argument it can be shown that due to the fact that S can be written as $\{B_S | i \in \{1, \ldots, n\}, \pi_i \in \Pi_i\}$, we can change summations and multiplication and get,

$$Pr(V = x_V) = \alpha'' \prod_{i=1}^{n} \left(\sum_{\pi_i \in \Pi_i} \gamma_{v_i}(v_i = x_i | \pi_i = x_{\pi_i}) \cdot 2^{m^B(v_i, \pi_i)} \right),$$

where $\gamma_{v_i}(v_i = x_i | \pi_i = x_{\pi_i})$ denotes the assessment function in the Bayesian belief network where v_i has parent set π_i. Now, let

$$\pi_i' = \cup_{\pi_i \in \Pi_i} \pi_i$$

for $i = 1, \ldots, n$, and let

$$\gamma_i'(v_i = x_i | \pi_i' = x_{\pi_i'}) = \alpha_i \cdot \sum_{\pi_i \in \Pi_i} \gamma_{v_i}(v_i = x_i | \pi_i = x_{\pi_i}) \cdot 2^{m^B(v_i, \pi_i)},$$

where α_i is a normalizing constant. Then $B_{S'}$ is a network structure with parent sets π_i, $i = 1, \ldots, n$ and γ'_i is an assessment function defining a joint probability distribution Pr over V. Note that the MDL measure can be considered to be an approximation of the Bayesian measure, this procedure applies when using the MDL quality of nodes instead of the Bayesian quality as well. Now, inference can be performed in this single Bayesian belief network B' instead of in the set of networks S.

When smoothing is used for learning a Bayesian belief network from data, appropriate sets of parent sets need to be found. We observe that for efficient reasoning with a Bayesian belief network, the parent sets should be as small as possible. Therefore, in applying the approach the parent sets selected for a node should have a large overlap. In most cases, not only the whole set of parents is informative for a node u, but also subsets are informative. The more informative a set of nodes for u, the higher the quality of this set. In practical applications, this can be achieved by taking the network structure generated by a search algorithm and letting Π_i be the set of subsets of π_i. Note that is likely that Π_i contains high quality parent sets.

To implement this approach, the qualities and the assessment functions of all subsets of all parent sets in the final structure need to be computed. Since this computation involves many terms that are also computed by the search algorithm, it is much more efficient to incorporate the estimation of the probabilities into the search algorithm. The basic idea is that every time an arc $v_j \to v_i$ is added to the network structure under construction the assessment function of v_i is adapted. Let π_i be the present parent set of v_i to which a node v_j is added and let $\gamma_{v_i}(v_i = x_i | \pi_i = x_{\pi_i})$ be the assessment function for v_i so far. Furthermore, let w_i be the sum of the contribution terms $2^{m(v_i, \pi_i)}$ of the parent sets considered so far. Then, the new probabilities $\gamma'_{v_i}(v_i = x_i | \pi_i = x_{\pi_i}, v_j = x_{v_j})$ are computed as the weighted sum $\alpha \cdot (w_i \cdot \gamma'_{v_i}(v_i = x_i | \pi_i = x_{\pi_i}) + 2^{m(v_i, \pi_i v_j)} \hat{P}r(v_i = x_i | \pi_i = x_{\pi_i}, v_j = x_{v_j}))$ where $\alpha = 1/(w_i + 2^{m(v_i, \pi_i v_j)})$ and $\hat{P}r$ a direct estimate; the new sum w_i of contributing terms is computed as $w_i + 2^{m(v_i, \pi_i v_j)}$.

Note that the sets Π_i consist of the subsets of the final parent sets of v_i that once were parent sets at some stage in the search algorithm. Also note that the quality of these parent sets are directly available in the algorithm. The only extra administration needed is maintaining the sums w_i of weights of parent sets so far. Details can be found in the algorithm, called *weighted K2*, shown below; a similar approach can be applied to algorithm B and we will call this algorithm *weighted B*. We would like to stress that this technique is suitable only for a greedy algorithm in which arcs are added one by one; when an arc $v_j \to v_i$ is deleted or reversed incremental computation of assessment functions would be much more complicated.

4.5 Experimental Results

We have performed several experiments to compare the performance of the various algorithms for learning Bayesian belief networks from data described in the previous sections. To this end, we generated Bayesian belief networks of different size, generated databases of cases from these networks, and applied the various learning algorithms to these databases using the original networks as golden standard to

```
**Algorithm weighted K2** $(<_V)$

for $i = 1, \ldots, n$ do            {initialize}
    $\pi_i \leftarrow \emptyset$
    $w_i \leftarrow 2^{m(v_i,\emptyset)}$
    for $x_i \in \Omega_{v_i}$ do $\gamma_{v_i}(v_i = x_i) = \hat{Pr}(v_i = x_i)$
for $i = 2, \ldots, n$ do            {main loop}
    repeat
        select $v \in \{v_1, \ldots, v_{i-1}\} \backslash \pi_i$ that
          maximizes $g = m(v_i, \pi_i \cup \{v\})$
        $\Delta \leftarrow g \Leftrightarrow m(v_i, \pi_i)$
        if $\Delta > 0$ then
            for $x_i \in \Omega_{v_i}$, $x_{\pi_i} \in \Omega_{\pi_i}$, $x_{v_j} \in \Omega_{v_j}$ do
                $\gamma_{v_i}(v_i = x_i | \pi_i = x_{\pi_i}, v_j = x_{v_j}) = \alpha(w_i \gamma_{v_i}(v_i = x_i | \pi_i = x_{\pi_i})$
                        $+ 2^g \hat{Pr}(v_i = x_i | \pi_i = x_{\pi_i}, v_j = x_{v_j}))$
            $w_i \leftarrow w_i + 2^g$
            $\pi_i \leftarrow \pi_i \cup \{v\}$
    until $\Delta \leq 0$ or $\pi_i = \{v_1, \ldots, v_{i-1}\}$
normalize $\gamma$
return $\gamma$
```

evaluate the performance of the algorithms.

We generated several Bayesian belief networks with varying numbers of nodes and arcs; details are listed in Table 4.2. We generated networks with 10, 15, and 50 nodes, respectively. We did not generate networks with more than 50 nodes for several reasons. First, we feel that in real-life domains there are not many databases with many more variables of sufficient size. Secondly, the complexity of the described search algorithms is quadratic in the number of nodes which prohibits experimentation with larger networks from a computational point of view. The numbers of arcs are chosen so as to yield a singly connected structure, a structure with about one and a half times the number of nodes, and a structure with twice as many arcs as nodes. In the last case, the generated network structures contain many cycles and large parent sets. These ratios of number of arcs and number of nodes seem realistic. For example, the ALARM network [5] contains 37 nodes and 46 arcs. In networks where a time component is involved, every variable tends to be dependent of its value in a previous time slot and a ratio of the number of arcs and number of nodes near two is easily obtained. However, depending on the domain, this ratio may vary considerably.

A single Bayesian belief network was synthesized as follows. First, the networks' structure was generated. Initially, an ordering on the variables was fixed. Then, two nodes u and v were selected randomly, and an arc $v \rightarrow u$ was added to the structure in the making if $v > u$ and $u \rightarrow v$, otherwise. In each successive step of the generation process, one of the variables that already had at least one incident arc was selected and one of the variables with no incident arcs, both in random fashion. An arc was added between these nodes in the structure, taking the variable

# nodes	# arcs		
10	9	15	20
25	24	37	50
50	49	75	100

Table 4.2: The numbers of nodes and arcs in generated networks.

ordering into consideration. This process was repeated until a poly-tree resulted. If the desired number of arcs was not yet reached, randomly two different nodes were selected and an arc was added between them, once more taking the variable ordering into consideration. This step was repeated until the network structure contained the desired number of arcs. We would like to note that this generation process yields network structure with a bias towards structures with nodes having many incident arcs as opposed to networks with long strings of nodes; realistic networks seem to have the same kind of bias. For a generated network structure, assessment functions were generated for all nodes. The variables were all assumed to be binary. For each variable u with parent set π_u, we selected a random number from the unit interval and assigned it to $\gamma_u(u = 0|\pi_u = x_{\pi_u})$, setting $\gamma_u(u = 1|\pi_u = x_{\pi_u}) = 1 \Leftrightarrow \gamma_u(u = 0|\pi_u = x_{\pi_u})$, for each $x_{\pi_u} \in \Omega_{\pi_u}$. In this way, we generated ten different Bayesian belief networks for each combination of the number of nodes and number of arcs, ninety in total.

From each Bayesian belief network, we generated a database for each of the following sizes: 100, 200, 300, 400, and 500 cases. For this purpose, we used logic sampling [55] which we describe in further detail in the next chapter.

The performance of the various algorithms was evaluated with different criteria depending on the purpose of application of Bayesian belief network learning. When a network structure will be used as a starting point for a build-test cycle, it is important that as many as possible dependencies in the domain are represented in the structure. The number of extra and missing edges in the underlying graph of the learned network structure compared to the original structure therefore is used as a criterion for evaluating the performance of learning algorithms. When learning is used to recover causality, the direction of the arcs in a network is of importance. The number of extra and missing arcs in the learned network structure compared to the original structure therefore is used as another criterion. Consider the numbers of extra and missing edges in the underlying structure of the network structure yielded upon application of the K2 algorithm. Note that these numbers are equal to the numbers of extra and missing arcs if the ordering used for K2 is the same as the ordering used for constructing the network structure. For many real life applications involving decision making, it is important that the probability distribution represented by a Bayesian belief network approximated the true probability distribution. Therefore, the third criterion we used for evaluating the performance of the various algorithms for learning belief networks from data is the expected logarithmic distance between the learned distribution and the original distribution known as the *divergence* or *cross entropy*. The divergence is an appropriate criterion for this purpose as it emphasizes deviations for small probabilities for which errors have a large influence on decision making processes. The divergence is defined as

$$\sum_{x_V \in \Omega_V} Pr(V = x_V) \cdot \log \frac{Pr(V = x_V)}{Pr_B(V = x_V)},$$

where Pr is the 'true' probability distribution, that, is the distribution represented by the Bayesian belief network that was used to generated the database D, and Pr_B denotes the distribution represented by the Bayesian belief network learned from D. We implemented the divergence by brute force. Because of computational limitations, we only measured the divergence for Bayesian belief networks with ten nodes. When the network structure that fits the database best is searched for, as for example in hypothesis testing, the quality of a structure is crucial. The last criterion used therefore is the quality of the returned network structure.

We applied the learning algorithms to the generated databases recording the number of extra and missing arcs, the number of extra and missing edges, the quality of the returned network structure and the divergence of the represented distribution. All these methods of evaluation were considered with respect to the original Bayesian belief network with which the database was generated.

Recall that we are interested in the performance of the various quality measures we discussed: the Bayesian measure, the information criterion and the MDL measure. For the information criterion, we experimented with the values $1/3$, $1/2$, 1, 2, and 3 that are often mentioned in literature; we will denote the criterion with these values by ic,1/3, ic,1/2, ic,1, ic,2, and ic,3, respectively.

We divided the experiments into two groups; experiments involving learning with greedy search heuristics, and experiments involving learning with general search heuristics. Learning assessment functions was addressed in both groups. In applying a quality measure, we assumed that there was no prior information. So, the probability distribution over all network structures was taken to be uniform.

All experiments were performed on a HP-9000 series 700 using a C-program.

4.5.1 Results for Greedy Search Heuristics

In this section, we illustrate the main features of the experimental results by considering small parts of the experimental results. Since the amount of experimental results is very large, we do not discuss all of it in detail.

We performed an experiment where we applied the ordinary K2 algorithm for Bayesian belief networks with ten nodes and nine arcs where the ordering on the nodes provided was the ordering used for constructing the original network structure. Figure 4.8 shows the number of extra arcs on the left-hand side and number of missing arcs on the right-hand side for various database sizes and quality measures averaged over ten networks.

From Figure 4.8, it is seen that the number of extra arcs as well as the number of missing arcs decreases when the database size grows. This effect is explained by the fact that the larger a database, the more information is available. So, the larger the database, the better the returned network structure. The figure also indicates that the network structure learned with the MDL measure tends to contain less extra arcs and more missing arcs than those learned with the Bayesian measure. This

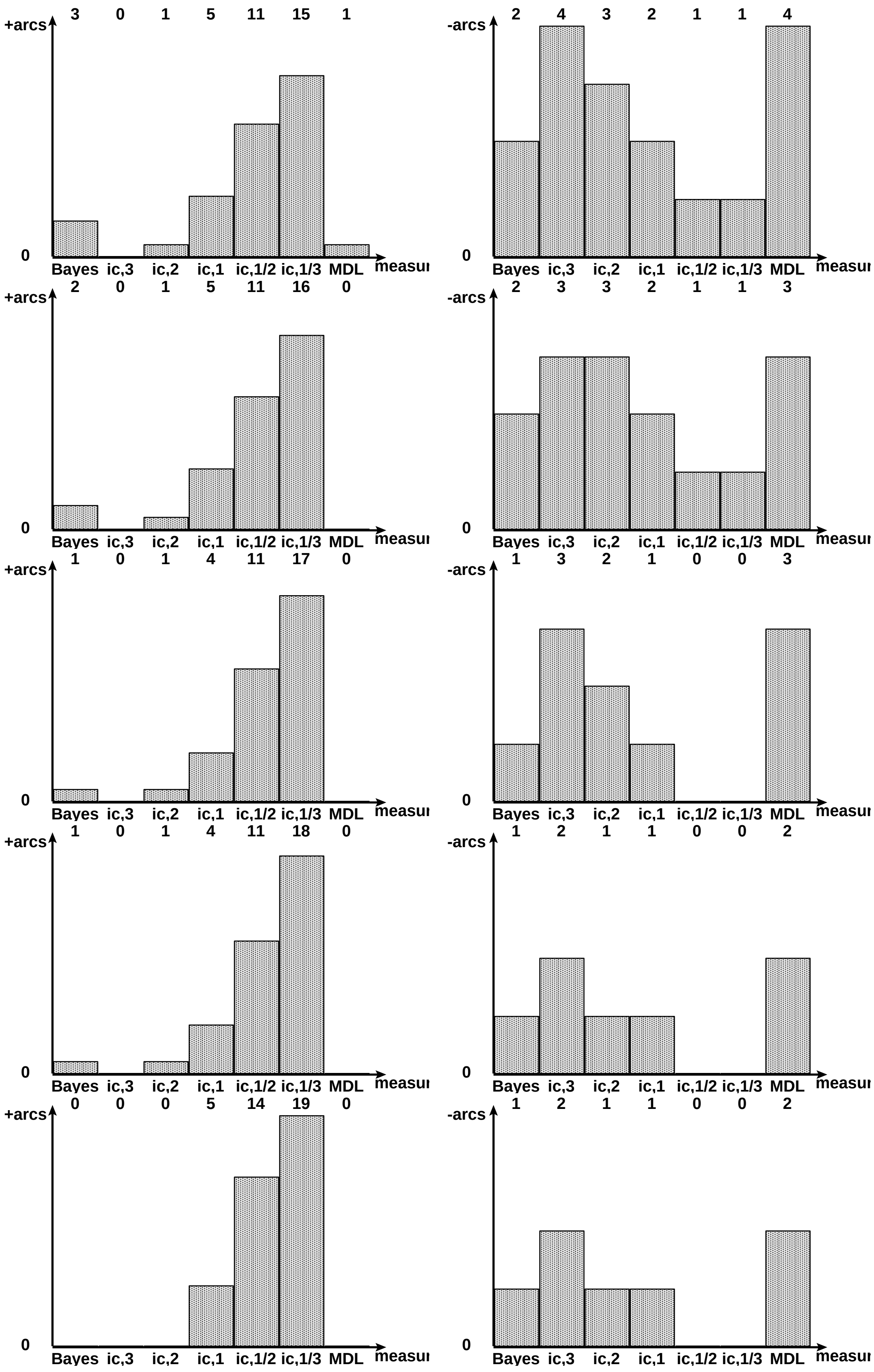


Figure 4.8: Average number of extra arcs (left) and missing arcs (right) obtained with ordinary K2 for various quality measures for databases of 100 (top) up to 500 (bottom) cases. The original network structures contained 10 nodes, and 9 arcs.

effect is explained by the observation that network structures returned when using the MDL measure tends to contain smaller parent sets than the structures returned when using the Bayesian measure. We described this effect from a theoretical point of view in Section 4.2.7.

When we examine the results for the information criteria with various penalty functions, we see that a decrease in the penalty function results in an increase in the number of extra arcs and a decrease in the number of missing arcs. Since, this function can be interpreted as a penalty for the number of values that need to be estimated for the assessment functions of a network structure, it will be evident that the higher this penalty, the smaller the sizes of the parent sets, and hence, the smaller the number of extra arcs, and the higher the number of missing arcs.

Table 4.3 shows the divergence averaged over ten networks between the learned distribution and the original one for Bayesian belief networks learned, where the original networks had ten nodes and nine arcs learned with algorithm B with one-step look-ahead. The same but diminished effects can be observed in the results with K2 with one-step look-ahead. In Table 4.3, the averages and variances of the divergence of the ten learned networks both for direct and smoothed estimates are listed. Note that with a growing number of extra arcs, a growing number of values need to be estimated for the various assessment functions. It will be evident that the more values need to be estimated, the more errors are introduced. On the other hand, the more missing arcs in a network, the fewer dependencies are represented and the more the learned probability distribution differs from the original one. The experimental results shown in Table 4.3 suggest that the former effect has more impact on the divergence than the latter effect. In fact, the Bayesian belief networks with the least arcs, namely the ones learned with the information criterion with penalty function $f(N) = 3$, have the lowest divergence of all. Table 4.3 further shows that applying the method of smoothing slightly decreases both the divergence and the variance of the divergence.

One would expect that the divergence would decrease with increasing database sizes. However, this effect cannot be observed from Table 4.3 for all but the Bayesian measure. This can be explained by observing that the error made in estimating the distribution tends to be rather larger due to the fact that the databases are relatively small. For accurate estimates, the database should contain several thousands of cases.

Figure 4.9 gives an impression of the usefulness of applying a k step look-ahead in the greedy search algorithm described in Section 4.3.2. The quality averaged over ten networks is shown for databases with 500 cases and the quality measure used was the MDL measure. The qualities are depicted for the algorithms K2 and B with one, two, and three-steps look-ahead, respectively. The table shows the qualities for network structure over ten nodes with 9, 15 and 20 arcs. In general, the variants of K2 returned a network structure with a slightly lower quality than with B. This effect can be explained from the observation that the ordering with which the network structure was constructed was supplied to K2. This ordering gives extra information about the original structure that was not available to B. For the network structures with 9 arcs, applying a look-ahead of more than one step with K2 did not influence the quality of the returned structure. However, for network structures with 15 and 20 arcs, the quality of the returned structure

Quality		Direct estimate		Smoothed	
measure	N	average	variance	average	variance
Bayes	100	1.3573	0.807853	1.3343	0.814999
	200	1.054	1.18945	1.0498	1.11725
	300	0.9613	0.984862	0.9466	0.976641
	400	0.8162	0.543504	0.8094	0.546175
	500	0.83	0.593212	0.8215	0.586496
ic,3	100	0.4512	0.371678	0.4541	0.369206
	200	0.4136	0.508621	0.4173	0.511758
	300	0.2892	0.323984	0.2984	0.30815
	400	0.7417	0.541098	0.7164	0.535142
	500	1.2079	1.22244	1.2014	1.22201
ic,2	100	0.5247	0.354532	0.5198	0.334052
	200	0.6418	0.560836	0.622	0.555777
	300	0.4806	0.312307	0.4836	0.326394
	400	0.882	0.499015	0.8577	0.494539
	500	1.1217	0.738958	1.097	0.752403
ic,1	100	1.6027	0.730721	1.5151	0.715809
	200	1.6836	1.05287	1.5661	1.01165
	300	1.6485	0.727109	1.5207	0.6034
	400	1.5914	0.949219	1.5312	0.936412
	500	1.6706	0.925989	1.5921	0.933069
ic,1/2	100	1.7951	0.778958	1.8037	0.880254
	200	2.2496	1.11739	2.1666	1.09767
	300	2.5577	1.30925	2.4166	1.12787
	400	1.9844	0.729641	1.9567	0.677669
	500	2.1396	0.99537	2.0981	1.00064
ic,1/3	100	2.0171	0.648805	2.0336	0.779539
	200	2.5019	1.62503	2.3644	1.52051
	300	1.9755	0.808016	1.906	0.847741
	400	2.2598	0.790407	2.1673	0.790488
	500	2.5443	1.20251	2.5512	1.23562
MDL	100	0.6911	0.654127	0.6655	0.621631
	200	0.418	0.522918	0.4161	0.522632
	300	0.2892	0.323984	0.2949	0.310366
	400	0.7408	0.542232	0.7163	0.53562
	500	1.1034	1.26881	1.1019	1.26847

Table 4.3: Divergence for Bayesian belief networks with ten nodes and nine arcs learned by B.

increased considerably upon applying a two or three-step look-ahead. This effect can be explained from the observation that when the original network structure is sparse, the most influential nodes for inclusion in a parent set are easily discerned; when the original network structure is not sparse, there are relatively many influential nodes that can be selected for inclusion in a parent set. A more elaborate search explores the search space more effectively in such cases. Judging from Figure 4.9, it does not pay to have a three-step look-ahead instead of a two-step look-ahead: the quality of the returned network structure does not improve significantly while the run time of the algorithm increased considerably.

When algorithm B is applied with a two-step instead of a one-step look-ahead, network structures of lower quality were returned for networks with 9 and 20 arcs and higher quality for networks with 15 arcs. However, the increase in quality was much less impressive than for K2. When a three step look-ahead is applied, the quality of the returned network structures decreases a lot. The reason lies in our restriction on the neighborhood structure that the addition of up to three arcs is such that all arcs point to the same node. Therefore, early in the execution of the algorithm, when a set of three arcs all pointing to the same node is chosen, we get severe restrictions on the ordering of the variables, which we do not get when the arcs are added one by one. This effect also appears with the two-step look-ahead, but it is compensated for by the more effective search for parents, resulting in a slightly better quality.

To see whether the aforementioned effects are also present with larger networks, we performed the same experiments as above, but now for belief networks with 25 nodes. To keep computational time limited, we did not apply the greedy search heuristic with a three-step look-ahead, but only the variants with one and two-step look-ahead. Figure 4.10 shows the average quality over ten networks for which databases with 500 cases were generated and where the MDL measure was used. Again, we see that in most cases, K2 with one-step look-ahead performs slightly better than B with one-step look-ahead.

Further, when a two-step look-ahead is used, the average quality of the returned network structures is the same when the original network structure has the topology of a poly-tree. However for less sparse network structures, there is a significant increase in quality when a two-step look-ahead is used instead of a one-step look-ahead. These effects also have been observed in the experiments with the network structures containing ten nodes. A short examination of other outcomes shows that the same behavior is exposed as for the network structures with ten nodes. Only the numbers increase dramatically for larger networks. For example, with a Bayesian measure, network structures with up to 200 extra arcs were returned for networks with 50 nodes and with the MDL measure, network structures with up to 40 missing arcs were constructed. In general, the experiments with larger networks showed the same effects we have discussed before for the various evaluation methods, but now more dramatically. So, it seems sufficient to perform experiments on small networks to get an impression of the various learning methods in general.

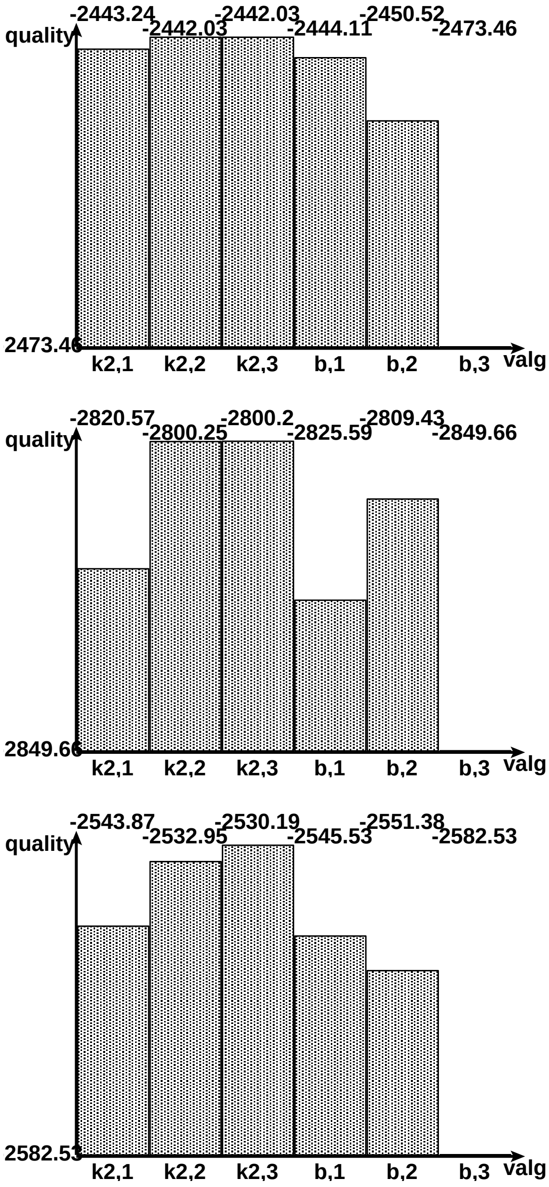


Figure 4.9: Average quality for various algorithms and number of arcs with original network structure containing 9 (top), 15 (middle), and 20 (bottom) arcs. The databases were over ten nodes and had size 500 and network structure learned using the MDL measure.

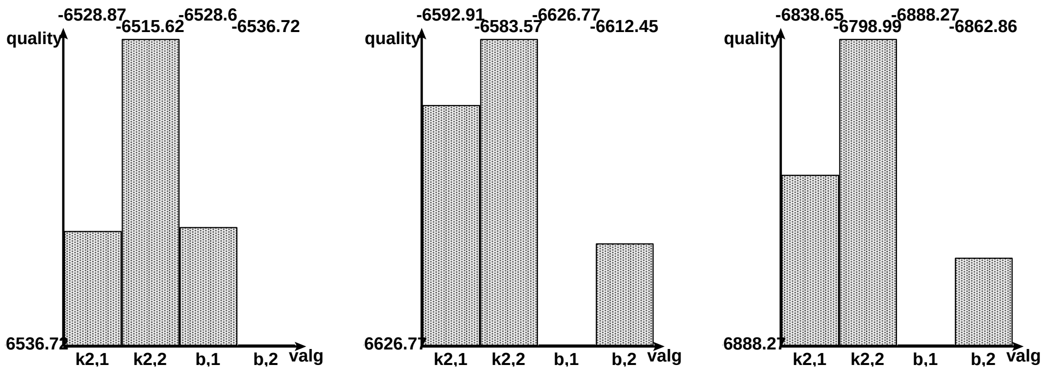


Figure 4.10: Average quality for various algorithms and number of arcs with original network structure containing 24 (left), 37 (middle), and 50 (right) arcs. The databases were over ten nodes and had size 500 and network structure learned using the MDL measure.

4.5.2 Results for General Search Heuristics

In this section, we illustrate the performance of the general search algorithms discussed in Section 4.3.3. In our experiments, we investigated the behavior of these algorithms only for the Bayesian belief networks with 10 variables and 15 arcs and the databases generated from them; we do not expect other trends to arise from experiments with larger networks, supported by the results obtained from the experiments with the greedy search heuristics. We considered the influence of the various parameters on tabu search, simulated annealing and rejectionfree annealing separately and compare their performance with optimal parameters with the greedy search heuristics. Also, to keep computational time limited, we used the small domains. For all general search algorithms, we performed the experiments with both the straight and the reversed neighborhood structure. We executed every algorithm with 100, 250, 500, 1000, 2500, 5000, 7500, and 10000 iterations, which seems a broad enough spectrum of iterations to elucidate the various properties of the algorithms. Due to computational limitations, we did not perform experiments with a larger number of iterations. For each algorithm, we further experimented with various parameter settings.

Tabu Search

Table 4.4 lists the quality averaged over ten networks of the network structures learned by the tabu-search algorithm with the Bayesian measure with 100 up to 10000 iterations for databases with 500 cases generated using network structures with 15 arcs. The parameter that we can select freely for tabu search is the length of the tabu-list. In literature, one encounters values between 20 and 50. So, we performed experiments with a tabu-list length of 20, 30 and 50. The part on the left side of the table shows the quality averaged over ten networks of the learned structures when tabu search is employed using a straight neighborhood structure; the part on the right side of the table shows the average quality when the reversed neighborhood structure is used. It seems that by using the reversed structure, the

	straight			reversed		
	neighborhood structure			neighborhood structure		
ni	20	30	50	20	30	50
100	-2767.84	-2765.85	-2772.16	-2760.91	-2762.32	-2764.35
250	-2766.74	-2765.85	-2768.60	-2759.85	-2760.30	-2761.79
500	-2766.64	-2765.85	-2767.09	-2759.80	-2760.30	-2761.79
750	-2766.60	-2765.85	-2769.44	-2759.80	-2760.14	-2761.73
1000	-2766.60	-2765.85	-2771.09	-2759.80	-2759.98	-2761.55
2500	-2766.60	-2765.85	-2769.67	-2759.72	-2759.84	-2761.19
5000	-2766.60	-2765.68	-2769.15	-2759.72	-2759.84	-2760.78
7500	-2766.47	-2765.37	-2768.81	-2759.68	-2759.83	-2760.49
10000	-2766.46	-2765.37	-2767.65	-2759.67	-2759.83	-2760.41

Table 4.4: Average quality of the network structure learned by tabu search with the Bayesian measure for different neighborhood structures for networks with 10 nodes and 15 arcs.

search space of network structures is explored more effectively since then the average quality of the learned network structure is considerably better than by using the straight neighborhood structure. This effect is explained by the observation that reversal of an arc is less often performed when using the straight neighborhood structure because removal of the arc first results in a network structure with a relatively low quality. A similar observation holds if the performance of tabu search is evaluated using the number of extra and missing edges and arcs and less dramatically when using the divergence. In the experiments that we present from now on, we have used the reversed neighborhood structure.

We want to know the influence of the two parameters tabu-list length and number of iterations on the performance of tabu search. In Table 4.5, the number of missing and extra edges are listed for tabu search using the reversed neighborhood structure averaged over ten networks. The part on the left of the table shows the number of missing edges for various tabu-list lengths; the part on the right of the table shows the number of extra edges. The rows of the table indicate the results for a different number of iterations. The table shows that with a tabu-list of length 20 slightly better results are obtained than with tabu-lists of length 30 and 50. However, in experiments with networks with other numbers of arcs and with other database sizes we obtained other results. Therefore, we conclude that the tabu-list length hardly influences the performance of tabu search.

Furthermore, we see in Table 4.5 that with an increasing number of iterations, the average number of missing and extra arcs decreases, especially for a tabu-list length of 50. This effect can be explained by observing that with a larger number of iterations a larger part of the search space is visited by the algorithm. On the other hand, it seems that not much progress is made after 2500 iterations have been made. The same observations hold for Table 4.4.

When the number of extra and missing arcs was used to evaluate the performance of tabu search, we obtained similar tables as Table 4.5. Only the numbers were somewhat higher due to having reversed arcs in the learned network counting both

	Missing edges			Extra edges		
ni	20	30	50	20	30	50
100	1.1	1.1	1.5	2.0	2.7	2.7
250	0.9	0.9	1.5	1.5	2.1	2.0
500	0.9	0.9	1.5	1.3	2.1	2.0
750	0.9	0.9	1.3	1.4	1.7	2.1
1000	0.9	0.7	1.1	1.4	1.5	2.1
2500	0.8	0.9	1.2	1.4	1.6	1.8
5000	0.8	0.9	1.0	1.4	1.6	1.9
7500	0.7	0.9	1.1	1.4	1.6	1.9
10000	0.8	0.9	1.1	1.5	1.6	1.7

Table 4.5: Average number of missing and extra edges of network structures returned by tabu search learned with the Bayesian measure for networks with 10 nodes and 15 arcs.

as missing and as extra arcs. The divergence of the networks obtained with tabu search was comparable to those obtained when using the greedy heuristics; the divergence was hardly influenced by the number of iterations and tabu-list length.

When we compare these results with the experiments with the greedy search heuristics, we see that K2 returned networks with an average quality of 2774.2 and 2761.61 for one and two-step look-ahead respectively when using the same databases as above. From Table 4.4 we conclude that a gain in quality is obtained by using tabu search with respect to the greedy search heuristic K2.

For K2 we found on average of 2.5 and 1.0 missing edges for one and two-step look-ahead respectively and zero extra edges for both cases. So, it seems that K2 performs better than tabu search when we consider the number of mismatched edges, that is, the sum of missing and extra edges. To conclude, we would like to note that the runtime of K2 is considerably less than that of tabu search: K2 took on average 0.37 seconds for networks with 10 nodes and 15 arcs databases with 500 cases, which was the worst case. In comparison, tabu search took 1.46 seconds for 100 iterations in the experiments above described, and the runtime grows linearly in the number of iterations to 139.13 seconds for 10.000 iterations.

The divergence of the returned networks was comparable for both the greedy heuristics and tabu search.

Simulated Annealing and Rejectionfree Annealing

For the simulated annealing and rejectionfree annealing algorithms, Table 4.6 lists the quality averaged over ten networks of the network structures learned from the same databases as before for various numbers of iterations. The simulated annealing algorithm depends on a cooling schedule. All experiments with the simulated annealing algorithm were performed by starting with a temperature of $0.5 \cdot no$ where no is the number of cases in the database and a final temperature of $0.025 \cdot no$. For both algorithms we see that with an increasing number of iterations the average quality shows a tendency to become of higher quality. This observation can be explained

ni	simulated annealing	rejectionfree annealing
100	-2799.22	-2767.07
250	-2781.45	-2760.28
500	-2770.18	-2760.98
750	-2766.48	-2761.57
1000	-2765.99	-2759.59
2500	-2762.39	-2760.40
5000	-2760.56	-2759.58
7500	-2760.43	-2759.58
10000	-2759.90	-2759.80

Table 4.6: Average quality of network structures returned by simulated annealing and rejectionfree annealing with the Bayesian measure for networks with 10 nodes and 15 arcs.

by the fact that with an increasing number of iterations a larger part of the space of network structures is searched.

Table 4.7 shows the number of missing and extra edges averaged over ten networks for the same databases as above. On the left side of the table, the average number of missing edges are listed. The first column for network structures returned by simulated annealing and the second for rejectionfree annealing. Likewise, the right-hand side shows the average number of extra edges. Each row show the results for a different number of iterations. Again we see that the more iterations are used, the better the returned network structures. Rejectionfree annealing seems to return network structures with a slightly lower number of extra and missing edges then simulated annealing does. Similar effects can be observed when considering the number of extra and missing arcs.

In both Table 4.6 and 4.7 we see that rejectionfree annealing performed better than simulated annealing. This observation can be explained by interpreting rejectionfree annealing as an efficient implementation of the last part of the cooling schedule in simulated annealing.

The divergence between the original networks and the networks learned with simulated and rejectionfree annealing was in general better than in the experiments with K2 when there were few arcs in the network structure and worse when there were many arcs.

4.5.3 Discussion

Limited value should be assigned to the experiments since relatively few networks were considered. Further, the networks considered were randomly generated and need not correspond to the networks encountered in particular real-world domains.

From the experiments, we learn that the choice of the quality measure used for learning Bayesian belief networks depends on the use of the learned Bayesian belief network or network structure. When it is used as start of a build-test cycle, it may be better to have a network structure that represents all dependencies. Then a domain

	Missing edges		Extra edges	
ni	simulated annealing	rejectionfree annealing	simulated annealing	rejectionfree annealing
100	2.6	1.9	9.0	2.8
250	1.9	1.3	6.2	2.2
500	1.7	1.6	4.6	2.5
750	1.4	1.2	3.9	2.2
1000	2.1	1.0	4.5	1.8
2500	1.3	1.3	2.6	1.9
5000	1.2	1.2	2.4	1.9
7500	1.3	1.2	2.0	1.9
10000	1.0	1.4	2.1	2.1

Table 4.7: Average number of missing and extra edges of network structures returned by simulated annealing and rejectionfree annealing.

expert only needs to identify the suggested dependencies that are not significant or that arose by coincidental occurrences of cases. In this case, it is better to use an information criterion with a small penalty function or the Bayesian measure. When the learned Bayesian belief network is used for classification, a good approximation of the distribution over the domain is important. In this case, the best quality measure is an information criterion with a very high penalty function. When the results need to be interpreted in terms of probabilities, the Bayesian measure may be used. Alternatively, the MDL measure may be interpreted as approximation of the Bayesian measure.

In general, the performance differences of the search algorithms is not very large. Therefore, we believe that it is not worth the effort to investigate these general purpose algorithms or to develop more clever heuristics. The choice of the search algorithm also depends on the use of the learned network. If computation time is important, for example, when a network is evaluated every time it has been used for inference, then a greedy search heuristic is most suitable. If the quality of the distribution is important, one may select a greedy search heuristic when there turn out to be many arcs in the network structure, and otherwise the general search heuristics perform fine. By running K2 with some good node ordering, one can determine with little computational effort whether the network structure contains many arcs or not. If the quality of the network structure is important, rejectionfree annealing is most suitable. If few missing and extra edges are required then tabu search seems to be the best choice of the general search heuristics. However, K2 performs comparable when a good ordering is available.

For learning assessment functions smoothing seems to give slightly smaller errors in the estimates of these functions than direct estimates do. However, large databases are necessary to obtain reliable estimates, especially when the network structure contains many arcs and many values need to be estimated. The database used in the experiments contained at most 500 cases. Databases of the used sizes are too small to obtain useful estimates.

4.6 Appendix

In this appendix, we show some additional properties of the Bayesian measure and we show that a probability distribution exists that induces the independency model used in the proof of Theorem 4.5.

We show some properties of the Bayesian measure for a database D_n of the form as defined in Section 4.2.7. In particular, we consider the parent set of node w in network structures learned from D_n. The parent set that consists of a number of consecutive variables $\{v_{n-k}, v_{n-k+1}, \ldots, v_n\}$ will be preferred by the Bayesian measure over a set that consists of a number of consecutive variables of the same size that is lower numbered.

4.7 **Lemma** Let $V = \{v_1, \ldots, v_n, w\}$ be a set of variables and let D_n be a finite database over V as defined in Section 4.2.7. Let $\pi'_w = \{v_{j-k}, v_{j-k+1}, \ldots, v_j\}$ for some $k \geq 0$, $k < j < n$, and let $\pi_w = \{v_{n-k}, v_{n-k+1}, \ldots, v_n\}$. Then, $m^B(w, \pi'_w, D_n) < m^B(w, \pi_w, D_n)$. □

Proof: We will refer to the configuration of π'_w when all nodes have value 0 as the first configuration of π'_w; when all nodes have value 1 as the last configuration of π'_w. Note that by construction of D_n we have that the number of cases in D_n for which w is 0 and π_w takes any other configuration than the last one is 2. Now, by definition, we have

$$\exp(m^B(w, \pi'_w, D_n)) = \prod_{j'=1}^{q'} \frac{N'_{(n+1)j'0}!\,N'_{(n+1)j'1}!}{(N'_{(n+1)j'} + 1)!}.$$

Let a be the number of cases in D_n for which w is 0 and all nodes in π'_w have value 0. Let b be the number cases in D_n for which w is 0 and all nodes in π'_w have value 1. Let $+s$ represent the case that the number of cases with w is 0 is more, less, or equal to w is 1 for the first configuration of π'_w. So, $t \in \{\Leftrightarrow 2, 0, 2\}$. Further let $+r$ represent that there may be two cases more with w is 0 than with w is 1 for the last configuration. So, $s \in \{0, 2\}$. Then, we can write $\exp(m^B(w, \pi'_w, D_n))$ as

$$\frac{a!(a+s)!}{(2a+1+s)!} \left(\frac{2!}{3!}\right)^{k-1} \frac{b!(b+r)!}{(2b+1+r)!}. \tag{4.27}$$

Note that there are six ways to interpret Formula 4.27. Likewise, we can write $\exp(m^B(w, \pi_w, D_n))$ as

$$\left(\frac{2!}{3!}\right)^{k} \frac{(a+b+d \Leftrightarrow 2)!(a+b \Leftrightarrow d+s+r)!}{(2a+2b+1 \Leftrightarrow 2+s+r)!}, \tag{4.28}$$

where d is 0 or 2 representing the possibility that for the first configuration, the number of cases where w is 0 may differ from the number of cases where w is 1. We will show that Formula 4.28 is an upper bound of Formula 4.27 and thus that $m^B(w, \pi'_w, D_n) < m^B(w, \pi_w, D_n)$.

By writing $x^{\underline{y}}$ for $\frac{x!}{(x-y)!}$ and grouping terms, we can write Formula 4.28 as

$$\frac{a!(a+s)!}{(2a+1+s)!}\left(\frac{2!}{3!}\right)^{k-1}\left(\frac{2!}{3!}\frac{(a+b+d\Leftrightarrow 2)^{\underline{b+d-2}}(a+b\Leftrightarrow d+s+r)^{\underline{b-d+r}}}{(2a+2b\Leftrightarrow 1+s+r)^{\underline{2b-2+r}}}\right). \tag{4.29}$$

The first two terms are exactly the same as in Formula 4.27. So, we concentrate on the third term

$$\frac{(a+b+d\Leftrightarrow 2)^{\underline{b+d-2}}(a+b\Leftrightarrow d+s+r)^{\underline{b-d+r}}}{(2a+2b\Leftrightarrow 1+s+r)^{\underline{2b-2+r}}}\frac{2!}{3!}. \tag{4.30}$$

We proceed comparing by 'peeling' terms of the form $\frac{x\cdot y}{w\cdot z}$ from Formula 4.30 starting with as high as possible values of $x\cdot y$ and $w\cdot z$. We can write Formula 4.30 as

$$\frac{(a+b+d\Leftrightarrow 2)(a+b\Leftrightarrow d+s+r)}{(2a+2b\Leftrightarrow 1+s+r)(2a+2b\Leftrightarrow 2+s+r)} \cdot\frac{(a+b+d\Leftrightarrow 3)^{\underline{b+d-3}}(a+b\Leftrightarrow d\Leftrightarrow 1+s+r)^{\underline{b-d-1+r}}}{(2a+2b\Leftrightarrow 3+s+r)^{\underline{2b-4+r}}}\frac{2!}{3!}. \tag{4.31}$$

By inspection, we find that

$$\frac{(a+b+d\Leftrightarrow 2)(a+b\Leftrightarrow d+s+r)}{(2a+2b\Leftrightarrow 1+s+r)(2a+2b\Leftrightarrow 2+s+r)} > \frac{b(b+r)}{(2b+1+r)(2b+r)}.$$

So, Formula 4.31 is larger than,

$$\frac{b(b+r)}{(2b+1+r)(2b+r)}\frac{(a+b+d\Leftrightarrow 3)^{\underline{b+d-3}}(a+b\Leftrightarrow d\Leftrightarrow 1+s+r)^{\underline{b-d-1+r}}}{(2a+2b\Leftrightarrow 3+s+r)^{\underline{2b-4+r}}}\frac{2!}{3!}.$$

Note that the last part is of the same form as 4.30 but now with $b\Leftrightarrow 1$ instead of b. Therefore, we can repeat the previous derivation. After $b\Leftrightarrow 2$ times applying this derivation, we get as lower bound of Formula 4.31,

$$\frac{b^{\underline{b-2}}(b+r)^{\underline{b-2}}}{(2b+1+r)^{\underline{2b-4}}}\frac{(a+d\Leftrightarrow 2)^{\underline{d}}(a\Leftrightarrow d+s+r)^{\underline{-d+r}}}{(2a+3+s+r)^{\underline{2+r}}}\frac{2!}{3!}. \tag{4.32}$$

By inspection, we find that,

$$\frac{(a+d\Leftrightarrow 2)^{\underline{d}}(a\Leftrightarrow d+s+r)^{\underline{-d+r}}}{(2a+3+s+r)^{\underline{2+r}}}\frac{2!}{3!} > \frac{2!(2+r)!}{5+r!}.$$

Therefore, Formula 4.32 is larger than,

$$\frac{b^{\underline{b-2}}(b+r)^{\underline{b-2}}}{(2b+1+r)^{\underline{2b-4}}}\frac{2!(2+r)!}{(5+r!)},$$

which is equal to

$$\frac{b!(b+r)!}{(2b+1+r)!}.$$

We conclude that Formula 4.30 is larger than the third term of Formula 4.29. Therefore, Formula 4.28 is larger than Formula 4.27, which completes the proof. □

Another property of the Bayesian measure with respect to to the parent set of w for a database D_n is that the Bayesian measure will prefer a parent set of w where the nodes are a set of consecutive nodes starting with v_k downwards over a parent set with non-consecutive nodes with highest node v_k.

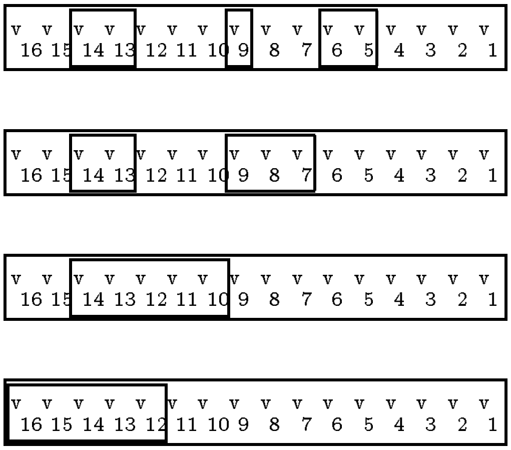


Figure 4.11: Example of parent sets of w

4.8 **Lemma** Let $V = \{v_1, \ldots, v_n, w\}$ and D_n be as before. Let $\pi_w = \{v_{n-k}, v_{n-k+1}, \ldots, v_n\}$ for some $0 \leq k\langle n$ and $\pi'_w \subset \{v_1, \ldots, v_n\}$, $|\pi'_w| = k+1$, $\pi'_w \neq \pi_w$. Then, $m^B(w, \pi'_w, D_n) < m^B(w, \pi_w, D_n)$. □

Proof: Regard parent set π'_w as a set of groups of consecutive nodes. In Figure 4.11, the upper parent set shows an example with four groups for D_{16}. Consider the quality of the parent set on top, and the one right below. In comparing their Bayesian measures, all configurations where $v_8 = 0$ need not be considered. And in fact, one could act as if D_8 was used.

By Lemma 4.7, we find that the parent set on top scores lower than the one right below. By the same argument, this parent set scores less than the one below it, and the one on the bottom scores highest of them all.

So, by shifting groups of nodes in the parent set we find parent sets that score better and better after each shift, where the parent set π_w has highest quality.

□

A third property of the Bayesian measure with respect to to the parent set of w for a database D_n is that the Bayesian measure will prefer a parent set of w where the nodes are a set of consecutive nodes starting with v_k downwards over a parent set obtained by removing the lowest numbered node from the previous parent set.

4.9 **Lemma** Let $V = \{v_1, \ldots, v_n, w\}$ and D_n be as before. Let $\pi'_w = \{v_{n-k}, v_{n-k+1}, \ldots, v_n\}$ for some $k < n \Leftrightarrow 1$ and let $\pi_w = \pi'_w v_{n-k-1}$. Then, $m^B(w, \pi'_w, D_n) < m^B(w, \pi_w, D_n)$. □

Proof:

By definition, π_w is equal to $\pi'_w v_{n-k-1}$. By definition, we have that $\exp(m^B(w, \pi'_w, D_n)) \Leftrightarrow \exp(m^B(w, \pi_w, D_n))$ is

$$\prod_{j'=1}^{q'} \frac{N'_{(n+1)j'0}! N'_{(n+1)j'1}!}{(N'_{(n+1)j'} + 1)!} \Leftrightarrow \prod_{j=1}^{q} \frac{N_{(n+1)j0}! N_{(n+1)j1}!}{(N_{(n+1)j} + 1)!}.$$

Let b be the number cases in D_n for which w is 0 and all nodes in π'_w are 1. Since all counts of configurations with $v_{n-k} = 0$ are the same, the above formula can be written as

4.33

$$C\left(\frac{b!(b+r)!}{(2b+1+r)!} \Leftrightarrow \frac{2!}{3!}\cdot\frac{b!(b \Leftrightarrow 2+r)!}{(2b \Leftrightarrow 2+r)!}\right),$$

where C is a positive constant. For $k < n \Leftrightarrow 2$, Formula 4.33 equals

$$C'\left(\frac{(b+r)(b \Leftrightarrow 1+r)}{(2b+1+r)(2b+r)} \Leftrightarrow \frac{2!}{3!}\right),$$

which by inspection is negative. So, for $k < n \Leftrightarrow 2$ the Bayesian measure prefers π_w over π'_w. For $k = n \Leftrightarrow 2$, Formula 4.33 equals

$$C'\left(\frac{1!3!}{5!} \Leftrightarrow \frac{2!}{3!}\frac{1!1!}{3!}\right) = C'\left(\frac{1}{20} \Leftrightarrow \frac{1}{18}\right),$$

which is negative as well. So, also for $k = n \Leftrightarrow 2$ the Bayesian measure prefers π_w over π'_w. □

A fourth property of the Bayesian measure with respect to to the parent set of w for a database, which summarizes the previous three properties, is that the Bayesian measure prefers the parent set of w containing all the other nodes over any other parent set.

4.10 **Lemma** Let $V = \{v_1, \ldots, v_n, w\}$ and D_n be as before. Let $\pi'_w \subsetneq \{v_1, \ldots, v_n\}$ and let $\pi_w = \{v_1, \ldots, v_n\}$. Then, $m^B(w, \pi'_w, D_n) < m^B(w, \pi_w, D_n)$. □

Proof: The property follows directly from the previous lemmas. For any $\pi'_w \subsetneq \{v_1, \ldots, v_n\}$ with $|\pi_w| = k \Leftrightarrow 1$ we have the parent set $\pi''_w = \{v_{n-k}, \ldots, v_n\}$ such that $m^B(w, \pi'_w, D_n) < m^B(w, \pi''_w, D_n)$ by Lemma 4.8. By repetitively applying Lemma 4.9, thus extending π''_w node by node, we find that $m^B(w, \pi''_w, D_n) < m^B(w, \pi_w, D_n)$. □

Now, we will show that a distribution exists that induces an independency model as used in the NP-completeness proof of Theorem 4.5. We show that the following distribution has this property.

4.12 **Definition** Let V be a set of binary variables. Let $G = (V, E)$ be an undirected graph with $|E| \geq 1$. Let $K : \mathbb{N} \rightarrow \mathbb{N}^+$ be defined as $K(n) = 2^{n^2}$. Let $par : \Omega_S \rightarrow \mathbb{N}^+$ be defined as $par_S(S = x_S) = K(|S|)$ if the number of ones in x_S is even, and $par_S(S = x_S) = 2 \cdot K(|S|)$ otherwise. Let S be the set of sets $\{B | B \subseteq V, \exists_{u,v \in B}(u, v) \in E\}$. Then, the *parity distribution* associated with G is the joint probability distribution on V defined as

$$Pr(V = x_V) = \alpha \cdot \sum_{B \in S} par_B(B = x_B),$$

where α is a normalization constant. □

Note that the graph G in the definition is not a network structure nor a Markov network. Furthermore, G has to have at least one edge since, otherwise $S = \emptyset$ and $Pr(V = x_V) = 0$ for all configurations of V, which would not define a proper distribution.

First, we show a property of marginalization of the parity distribution, then we consider the independency model induced by a parity distribution. The following lemma states that a marginalized parity distribution is a new parity distribution plus some constant.

4.11 **Lemma** Let V be a set of binary variables, and let G, S, par, and Pr be as in Definition 4.12. For any $R \subset V$, we have

$$Pr(V \backslash R = x_{V \backslash R}) = \alpha \cdot \sum_{B \in S, B \subseteq V \backslash R} par_B(B = x_B) + C_R,$$

where α is a normalization constant and C_R is a positive constant. □

Proof: We prove the property stated in the lemma by induction to the cardinality of R.

For $|R| = 0$, the property stated in the lemma trivially holds with $C_\emptyset = 0$.

Assume that for some $k \geq 0$ the property stated in the lemma holds for all set R with $|R| < k$. Now, let $|R| = k$. We consider $Pr(V \backslash R = x_{V \backslash R})$. Let $v \in R$, then by marginalization we have $Pr(V \backslash R = x_{V \backslash R}) = \sum_{x_v \in \Omega_v} Pr(v = x_v, V \backslash R = x_{V \backslash R})$. Since $Pr(v = x_v, V \backslash R = x_{V \backslash R})$ is the marginalization of $Pr(V = x_V)$ over $k \Leftrightarrow 1$ variables, we can apply the the induction hypothesis. So, we have

4.34
$$Pr(V \backslash R = x_{V \backslash R}) = \sum_{x_v \in \Omega_v} \alpha \cdot \sum_{B \in S, B \subseteq V \backslash (R \backslash v)} par_B(B = x_B) + C_{R \backslash v}.$$

Since α is not a function of x_v, Formula 4.34 can be written as $\alpha \cdot \sum_{x_v \in \Omega_v} \sum_{B \in S, B \subseteq V \backslash R} par_B(B = x_B) + C_{R \backslash v}$. By changing order of summation, we have $\alpha \cdot \sum_{B \in S, B \subseteq V \backslash R} \sum_{x_v \in \Omega_v} par_B(B = x_B) + C_{R \backslash v}$. When we split this sum into terms with sets that contain v and that do not and sum over v, we obtain

$$Pr(V \backslash R = x_{V \backslash R}) = \alpha \left\{ \sum_{B \in S, B \subseteq V \backslash R} (par_B(B = x_B) + par_B(B = x_B)) + \right.$$

$$\left. \sum_{B \in S, B \subseteq V \backslash (R \backslash v), v \in B} (par_B(B \backslash v = x_{B \backslash v}, v = 0) + par_B(B \backslash v = x_{B \backslash v}, v = 1)) \right\} + C_{R \backslash v}.$$

By grouping terms and by definition of par, we get that $Pr(V \backslash R = x_{V \backslash R})$ equals

$$\alpha \left\{ \sum_{B \in S, B \subseteq V \backslash Rv} 2 par_B(B = x_B) + \sum_{B \in S, B \subseteq V \backslash (R \backslash v), v \in B} (1 + 2).K(|B|) \right\} + C_{R \backslash v}.$$

Now, the last term within braces is not a function of the values of variables in V anymore. So, we can substitute a constant $C_R = 3\alpha \cdot \sum_{B \in S, B \subseteq V \backslash (R \backslash v), v \in B} K(|B|) +$

$C_{R\backslash v}$, and $\alpha' = 2\alpha$. Thus, we obtain $Pr(V\backslash R = x_{V\backslash R}) = \alpha' \cdot \sum_{B \in S, B \subseteq V\backslash R} par_B(B = x_B) + C_R$. □

Using the previous lemma, we show now $I(X, Z, Y)$ holds in a parity distribution induced by an undirected graph G if and only if for all $u, v \in XYZ$ there is no edge between u and v in G.

4.12 **Lemma** Let V be a set of binary variables, and let $G = (V, E)$ and Pr be as defined in Definition 4.12. Then, for any disjoint subsets $X, Y, Z \subseteq V$ $I(X, Z, Y)$ holds in Pr if and only if $\forall_{u,v \in XYZ, u \neq v}(u, v) \notin E$. □

Proof: We first show the *if* part of the lemma. Let X and Y be disjoint subsets of V such that $\forall_{u,v \in XY, u \neq v}(u, v) \notin E$. If $I(X, \emptyset, Y)$ can be shown, then also $I(X', Z', Y')$ for all $X' \subseteq X$, $Y' \subseteq Y$, $Z = XY\backslash X'Y'$ by the weak union and symmetry axioms. Now, $Pr(XY = x_{XY})$ is a marginalization of $Pr(V = x_V)$, which by Lemma 4.11 is,

$$\alpha \cdot \sum_{B \in S, B \subseteq XY} par_B(B = x_B) + C_{V\backslash XY}.$$

By construction of S there is no $B \in S$ such that $B \subseteq XY$ since there is no pair $u, v \in XY$ containing an edge. So, $Pr(XY = x_{XY})$ is constant $C_{V\backslash XY}$ for any configuration x_{XY} of XY. From $Pr(X = x_X) = \sum_Y Pr(XY = x_{XY})$ and $Pr(Y = x_Y) = \sum_X Pr(XY = x_{XY})$, we derive $Pr(X = x_X)Pr(Y = x_Y) = \sum_Y Pr(XY = x_{XY}) \sum_X Pr(XY = x_{XY})$. From the above argument, we find that $Pr(X = x_X)Pr(Y = x_Y)$ is $\sum_{x_Y \in \Omega_Y} C_{V\backslash XY} \sum_{x_X \in \Omega_X} C_{V\backslash XY}$. Summation over the constants gives $|X||Y|C_{V\backslash XY}C_{V\backslash XY}$ which is a constant C itself. Since both $Pr(XY = x_{XY}) = C_{V\backslash XY}$ and $Pr(X = x_X)Pr(Y = x_Y) = C$ define a proper distribution over XY, we have that $C_{V\backslash XY}$ must be equal to C and thus $Pr(XY = x_{XY}) = Pr(X = x_X)Pr(Y = x_Y)$ which by definition tells that $I(X, \emptyset, Y)$ holds in Pr.

We now show the *only if* part. Let X, Y, Z be subsets of V such that there exist $u, v \in XYZ, u \neq v$ with $(u, v) \in E$. Then,

$$Pr(XYZ = x_{XYZ}) = \alpha \cdot \sum_{B \in S, B \subseteq XYZ} par_B(B = x_B) + C_{V\backslash XYZ}. \tag{4.35}$$

By construction of S there is at least one set B, $B \in S, B \subseteq XYS$, namely $B = XYZ$. Let $m = |XYZ|$. Consider the contribution to Formula 4.35 of the sets smaller than m in S that are in XYZ; there are at most $m2^{m-1}$ such subsets and they contribute at most $2^{(m-1)^2}$ times 2. So, at most $m2^{m-1+(m-1)^2+1}$ is contributed. But the set XYZ contributes at least 2^{m^2} to Formula 4.35. We have,

$$m2^{m-1+(m-1)^2+1}/2^{m^2} = m2^{1-m} < 1,$$

for $m > 0$. Therefore, the 'parity' of XYZ dominates the summation and determines for a certain thresh hold level μ whether $Pr(XYZ = x_{XYZ}) > \mu$ or not. The 'parity' of a set cannot be written as the product of functions of two subsets. So, $I(X, Z, Y)$ does not hold in $Pr(V = x_V)$. □

Stratified Simulation

Inference is the main activity of knowledge-based systems: by means of inference, a knowledge-based system can reason with the knowledge represented in its formalism. For Bayesian belief networks, inference amounts to computing conditional probabilities for variables of interest, given the values of observed variables. Inference can be exact, that is, calculating exact probabilities, or approximate. Exact inference has been proven to be NP-hard [22]. So, it is not surprising that exact methods cannot be successfully used for every type of Bayesian belief network. The topology of a network structure has a large influence on the performance [108]. However, algorithms are available for exact inference[72, 81] that have a polynomial complexity for a wide range of network structures. In many applications, exact inference may not be necessary. For example, if the probability assessments in the network are not highly accurate, approximate probabilities can be delivered anyway. Also, it may suffice to know whether a certain probability has exceeded some threshold value and accuracy is not required. In such cases, approximate inference may be employed. Various algorithms have been developed [17, 55, 80, 97] for this task, most of which are based on the principle of simulation.

The basic idea of simulation is to generate a sample, that is, a multi-set of configurations using the distribution represented by the network. The probability of a variable taking a certain value is estimated from the sample by taking its relative frequency of appearance. It has been proven that approximating probabilities from a Bayesian belief networks with certain error properties is NP-hard [29]. Notwithstanding this result, approximate inference has many advantages over exact inference. The execution-time of an approximate method is linear in the number of nodes of the network n and the number of samples m (($O(n \cdot m)$). So, on fore-hand it is known what the execution-time of the algorithm will be. The execution-time of approximation algorithms hardly depends on the topology of the network. More specifically, no special mechanisms are needed for coping with loops. Another advantage of simulation algorithms is that they are easily parallelized; various processes can generate samples independently and they can then be combined into an estimate of the probabilities of interest. Simulation algorithms, further, have an ‘any-time’ property, that is, the algorithm may be stopped at any moment desired. Yet, the longer the sample generation is continued, the better the approximation of probabilities is.

However, observed values of variables tend to decrease the accuracy of approximations by simulation schemes. Many generated configurations may be very nonspecific for the observed evidence and only a small portion of the configurations may influence the estimates. As a result, the estimates are effectively based on only a few configurations, resulting in a large error in the estimation. Therefore, it is important that a simulation algorithm generates configurations evenly distributed over the sample space. Stratification is a popular statistical technique that can be exploited for this purpose. In this chapter, we present a new approximation algorithm based on stratification. With stratification, the generated configurations reflect the distribution of the Bayesian belief network better than with ordinary simulation techniques. Therefore, the error in the probability estimates is expected to be smaller. Efficient generation of samples using stratification is the topic of the present chapter.

In Section 5.1, we start with a general introduction of simulation and its application to inference in Bayesian belief networks. In Section 5.2, the stratified simulation algorithm is explained in detail and some variants are introduced. In Section 5.3, a theoretical analysis of the performance of these algorithms is given. In Section 5.4, experimental results are presented in order to get an impression of the performance of the stratified simulation algorithms compared to other approximation methods.

5.1 Simulation

Before turning to approximate inference for Bayesian belief networks, we give an informal introduction to the principles of simulation. Let $f : \mathcal{D} \rightarrow \mathrm{I\!R}$ be a real-valued function over a finite domain $\mathcal{D}$ called the *sample space*, and let

5.1 $$\phi = \sum_{x \in \mathcal{D}} \frac{f(x)}{|\mathcal{D}|}.$$

Now suppose that we want to find the value of ϕ without having to perform the summation over the entire sample space $\mathcal{D}$. Simulation is a technique for approximating the value of ϕ. At the basis of a simulation algorithm lies a *simulation scheme*. A simulation scheme is a procedure for selecting elements from the sample space. Each selected element is called a *sample trial*. A *sample* of size m is a multi-set of m sample trials. The idea of simulating f is to select a sample of trials $\{x_1, \ldots, x_m\}$, and to compute for each of the elements x_i from this sample the function value $f(x_i)$. The resulting multi-set $\{f(x_1), \ldots, f(x_m)\}$ is called a *simulation* of f. The mean μ of a simulation is an estimate for ϕ.

The simulation scheme used for generating a sample has a large influence on the quality of the estimate for ϕ. For example, when $f(x)$ is close to zero for almost all values x in the sample space $\mathcal{D}$ but very large for a small number of values, uniform selection of sample trials from $\mathcal{D}$ would lead to estimates for ϕ with a large error. A better way to select sample trials would then be to select sample trials proportional to their contribution $f(x)/m$ to the estimate for ϕ. More generally, a *sampling distribution* Pr_S over $\mathcal{D}$ is used to select sample trials. To compensate for the biased selection of sample trials with large values of f, the mean of a simulation is not calculated as the sum of the elements of the simulation but as a weighted sum over the sample probabilities. The so-called *sample score* of the simulation S is calculated as

5.2 $$\mu = \frac{\sum_{i=1}^{m} Pr_S(x = x_i)}{m} \sum_{i=1}^{m} \frac{f(x_i)}{Pr_S(x = x_i)}.$$

The contribution $\frac{f(x_i)}{Pr_S(x=x_i)}$ of sample trial x_i to the sample score μ is called the *trial score* of x_i, $i = 1, \ldots, m$. The law of large numbers now guarantees that μ converges to ϕ when the sample size m goes to infinity [92], that is,

5.3 $$\lim_{m \rightarrow \infty} \mu = \phi.$$

5.1.1 Simulation for Bayesian Belief Networks

Recall that approximate inference with a Bayesian belief network amounts to computing the estimates of (conditional) probabilities of interest from the network. Let $B = (B_S, B_P)$ be a Bayesian belief network over a set of variables V. Then, B defines a probability distribution [81]

5.4
$$Pr_B(V = x_V) = \prod_{u \in V} Pr_B(u = x_u | \pi_u = x_{\pi_u}),$$

where the conditional probabilities $Pr_B(u = x_u | \pi_u = x_{\pi_u})$ are defined by the assessment functions in B. Now, let X be a subset of V and suppose that we are interested in the probability of the configuration $X = x_X$. This probability is equal to

$$Pr(X = x_X) = \sum_{x_{V \backslash X} \in \Omega_{V \backslash X}} Pr_B(X = x_X, V \backslash X = x_{V \backslash X}).$$

Since most of the time we are interested in probabilities for single variables only, we will focus the discussion on single variables in the sequel. However, the presented theory is easily extended to apply to configurations that involve more than one variable.

When a subset of variables $E \subseteq V$ is known to adopt the configuration x_E for some $x_E \in \Omega_E$, then the probability $Pr_B(u = x_u | E = x_E)$ for some $u \in V \backslash E$ equals

5.5
$$Pr(u = x_u | E = x_E) = \sum_{x_{V \backslash uE} \in \Omega_{V \backslash uE}} Pr_B(u = x_u, V \backslash uE = x_{V \backslash uE} | E = x_E),$$

By definition, we have

$$Pr(u = x_u | E = x_E) = \sum_{x_{V \backslash uE} \in \Omega_{V \backslash uE}} \frac{Pr_B(u = x_u, V \backslash uE = x_{V \backslash uE}, E = x_E)}{Pr_B(E = x_E)},$$

which can be written as

5.6
$$Pr(u = x_u | E = x_E) = \alpha \sum_{x_{V \backslash uE} \in \Omega_{V \backslash uE}} Pr_B(u = x_u, V \backslash uE = x_{V \backslash uE}, E = x_E),$$

where α is a normalization constant such that the sum over all configurations $x_u \in \Omega_u$ of Formula 5.6 is one. The sum in Formula 5.6 can now be approximated by simulation, as described before. We do not simulate Formula 5.5 because $Pr_B(u = x_u, V \backslash uE = x_{V \backslash uE} | E = x_E)$ is not easily available. On the other hand, $Pr_B(u = x_u, V \backslash uE = x_{V \backslash uE}, E = x_E)$ is directly available from the assessment functions in the Bayesian belief networks. A sample trial then is a configuration of the set of variables $V \backslash E$.

Figure 5.1 shows the general framework of simulation algorithms for Bayesian belief networks. First, a simulation algorithm-dependent initialization step is performed. Then, m sample trials are generated from the network. Every sample trial is generated by assigning values to the variables of the network, either one by one or group-wise. In fact, there are different methods for generating sample trials; also

```
Initialize
$i \leftarrow 1$
while $i \leq m$ do
    $x_V \leftarrow$ generate configuration $i$
    $p \leftarrow Pr_B(V = x_V)/Pr_S(V = x_V)$
    update scores $p$
    $i \leftarrow i + 1$
Normalize scores
```

Figure 5.1: General simulation framework.

different sampling distributions are in use. The trial score p of a sample trial is calculated as the quotient of the value of the function that is being approximated, that is, the probability of the sample trial according to the distribution represented by the belief network $Pr_B(V = x_V)$, and the probability of generating this specific sample trial, that is, $Pr_S(V = x_V)$. With this trial score p, the sample scores for all values of variables that we are interested in are updated during the execution of a simulation algorithm; there are several methods available for updating the sample score. After the m trials have been generated, the scores are normalized such that the sum of approximate probabilities that a variable will take a value will be unity.

In summary, there are three components to a simulation algorithm for Bayesian belief networks:

1. a sampling distribution,
2. a sample trial generator, and
3. a scoring method.

We will review these components in the following subsections.

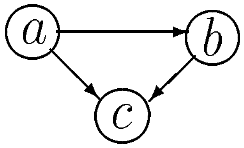
5.1.2 Sampling Distributions

The first component of a simulation algorithm for Bayesian belief networks that we consider is the sampling distribution. The sampling distribution defines a joint probability distribution over the nodes in the Bayesian belief network that are *sampled*. The sampling distributions used for sampling belief networks typically are distributions that can be written as the product of distributions over the sets of sampled nodes, that is, if U is the set of sets of nodes that are being sampled, then

$$Pr_S(V \backslash E = x_{V \backslash E}, E = x_E) = \prod_{W \in U} Pr_S(W = x_W | U_W = x_{U_W}), \qquad \textbf{5.7}$$

where each W is a *group* of nodes in U and U_W is a subset of V that is sampling distribution dependent. In most simulation algorithms, all nodes but the ones for which evidence has been obtained are sampled and the groups contain single nodes only. We distinguish between four different sampling distributions:

- the uniform distribution,



$$\gamma_a(a = 0) = 0.4$$

$$\gamma_b(b = 0|a = 0) = 0.6$$
$$\gamma_b(b = 0|a = 1) = 0.4$$

$$\begin{array}{ll}
\gamma_c(c = 0|a = 0, b = 0) = 0.3 & \gamma_c(c = 1|a = 0, b = 0) = 0.3 \\
\gamma_c(c = 0|a = 0, b = 1) = 0.3 & \gamma_c(c = 1|a = 0, b = 1) = 0.3 \\
\gamma_c(c = 0|a = 1, b = 0) = 0.3 & \gamma_c(c = 1|a = 1, b = 0) = 0.4 \\
\gamma_c(c = 0|a = 1, b = 1) = 0.3 & \gamma_c(c = 1|a = 1, b = 1) = 0.4
\end{array}$$

Figure 5.2: An example Bayesian belief network.

- the forward sampling distribution,
- the backward sampling distribution, and
- the Markov blanket distribution.

The *uniform distribution* [55] is used for sampling a single variable; it assigns the same probability to every value of the variable. So, for a variable u, we have that for all $x_u \in \Omega_u$

$$Pr_S(u = x_u) = \frac{1}{r_u},$$

where r_u is the number of values in Ω_u. Note that in terms of Formula 5.7 we have $W = u$ and $U_W = \emptyset$. Note that the sampling distribution is completely independent of the assessment functions in the Bayesian belief network at hand. Because no knowledge of the domain is incorporated, many non-representative sample trials are generated, that is, sample trials that have a negligible trial score and hardly contribute to the sampling score. Therefore, the uniform distribution is a sampling distribution that leads to unsatisfactory probability estimates.

Consider the Bayesian belief network shown in Figure 5.2 where the state spaces over the variables a, b, and c equal $\Omega_a = \Omega_b = \{0, 1\}$ and $\Omega_c = \{0, 1, 2\}$ respectively. The assessment functions associated with the variables are listed with the corresponding nodes; the probabilities of the configurations for which no assessment has been given are easily derived, for example, $\gamma_a(a = 1) = 1 \Leftrightarrow \gamma_a(a = 0) = 0.6$. Now suppose that node c is sampled with the uniform distribution. A random number generator is used to obtain a real value f between 0 and 1. If f is in the interval $[0..\frac{1}{3}]$, then the variable c is assigned the value 0; if f is in the interval $[\frac{1}{2}..\frac{2}{3}]$, then c is assigned the value 1, and if f is in the interval $[\frac{2}{3}..1]$ then c is assigned the value 2.

The *forward sampling distribution* [55] is also used for sampling a single variable. A variable, however, can only be forward sampled if all its parents have been assigned a value. The forward sampling distribution for a variable u is the distribution for u given the specific configuration of its parent set by the Bayesian belief network, that is, we have that

$$Pr_S(u = x_u|\pi_u = x_{\pi_u}) = \gamma_u(u = x_u|\pi_u = x_{\pi_u}),$$

for all $x_u \in \Omega_u$. Note that in terms of Formula 5.7 we have $W = u$ and $U_W = \pi_u$. In general, the forward sampling distribution results in good samples as long as the probability of the observed evidence is not very close to zero. If there is such evidence, the problem of generating many non-representative sample trials rises again.

Consider once more the Bayesian belief network from Figure 5.2. When the variables a and b in the network of have been assigned the values 0 and 1 respectively, node c is sampled by means of the assessment function for c. The probability that c will be assigned the value 0 is $Pr_S(c = 0|a = 0, b = 1) = 0.3$. Likewise, the probability that c will be assigned the value 1 and 2 is $Pr_S(c = 1|a = 0, b = 1) = 0.3$ and $Pr_S(c = 2|a = 0, b = 1) = 0.4$ respectively.

The *backward sampling distribution* [38] is used for sampling a set of nodes: the parent set π_u of a variable u can be sampled when u has been assigned a value. The configurations of π_u are assigned a probability according to the assessment function of node u. We have

$$Pr_S(\pi_u = x_{\pi_u}|u = x_u) = \frac{\gamma_u(u = x_u|\pi_u = x_{\pi_u})}{\alpha_u},$$

where α_u is a normalization constant. Note that in terms of Formula 5.7 we have $W = \pi_u$, and $U_W = u$. It has been suggested that the backward sampling distribution is much less sensitive to probabilities close to zero than the other distributions [38]. Therefore, sample trials that are generated with a backward sampling distribution may be more representative compared to sample trials generated with a forward sampling distribution when the probability of the evidence is very small. Since not all nodes can be backward sampled, backward sampling must be mixed with another sampling method like for example forward sampling.

To illustrate the use of the backward sampling distribution consider once more the Bayesian belief network from Figure 5.2. Suppose that the variable c has been assigned the value 1. Then, the probabilities with which a and b are assigned values are

$$\begin{array}{l} Pr_S(a = 0, b = 0|c = 1) \propto \gamma_c(c = 1|a = 0, b = 0) = 0.3, \\ Pr_S(a = 0, b = 1|c = 1) \propto \gamma_c(c = 1|a = 0, b = 1) = 0.3, \\ Pr_S(a = 1, b = 0|c = 1) \propto \gamma_c(c = 1|a = 1, b = 0) = 0.4, \text{ and} \\ Pr_S(a = 1, b = 1|c = 1) \propto \gamma_c(c = 1|a = 1, b = 1) = 0.4. \end{array}$$

The normalization constant equals $\alpha_c = 0.3 + 0.4 + 0.4 + 0.3 = 1.4$. The sampling distribution Pr_S therefore equals

$$\begin{array}{l} Pr_S(a = 0, b = 0|c = 1) = \frac{0.3}{1.4} = \frac{3}{14}, \\ Pr_S(a = 0, b = 1|c = 1) = \frac{0.4}{1.4} = \frac{2}{7}, \\ Pr_S(a = 1, b = 0|c = 1) = \frac{0.4}{1.4} = \frac{2}{7}, \text{ and} \\ Pr_S(a = 1, b = 1|c = 1) = \frac{0.3}{1.4} = \frac{3}{14}. \end{array}$$

The *Markov blanket sampling distribution* [80] is a distribution used for sampling a single variable u again. A variable, however, can only be sampled if all variables in its so-called Markov blanket have been assigned a value. Informally speaking, the *Markov blanket* M_u of a variable u is the set composed of all parents of u, the children of u, and the parents of the children of u except u itself. The values of node u are assigned a probability, that is, proportional to the product of appropriate values of the assessment functions of the nodes in its Markov blanket

M_u. The functionvalues are conform to the configuration $x_{V \backslash u}$ of the nodes $V \backslash u$ and the configuration x_u of u, that is, we have

$$Pr_S(u = x_u | V \backslash u = x_{V \backslash u}) = \alpha_u \gamma_u(u = x_u | \pi_u = x_{\pi_u}) \prod_{v \in M_u} \gamma_v(v = x_v | \pi_v = x_{\pi_v}),$$

where once more α_u is a normalization constant. Note that the values of the assessment functions that do not pertain to u itself are canceled out by the normalization constant. Therefore, we have that $Pr_S(u = x_u | V \backslash u = x_{V \backslash u})$ can be written as

$$Pr_S(u = x_u | M_u = x_{M_u}) = \alpha_u \gamma_u(v = x_v | \pi_v = x_{\pi_v}) \prod_{v \in \sigma_u} \gamma_v(v = x_v | \pi_v = x_{\pi_v}).$$

Note that in terms of Formula 5.7 we have that $W = u$ and $U_W = M_u$. Samples generated with the Markov blanket distribution give good approximations in general. However, if there is a strong dependence between variables it may happen that two strongly dependent variables keep each other at the same value in all sample trials, which may result in inaccurate estimates.

Suppose that we want to sample node b of the Bayesian belief network from Figure 5.2 with a Markov blanket sampling distribution. Further, suppose that the nodes a and c from the Markov boundary of b have been assigned the values $a = 0$ and $c = 1$ respectively. Then, the values of b are assigned probabilities

$$P_S(b = 0 | a = 0, c = 1) \propto \gamma_b(b = 0 | a = 0) \cdot \gamma_c(c = 1 | a = 0, b = 0) = 0.6 \cdot 0.3 = 0.18,$$

$$P_S(b = 1 | a = 0, c = 1) \propto \gamma_b(b = 1 | a = 0) \cdot \gamma_c(c = 1 | a = 0, b = 1) = 0.4 \cdot 0.4 = 0.16.$$

The normalization constant equals $\alpha_b = 0.18 + 0.16 = 0.34$. So, we have

$$P_S(b = 0 | a = 0, c = 1) = 0.18/0.34 = 9/17,$$

$$P_S(b = 1 | a = 0, c = 1) = 0.16/0.34 = 8/17.$$

The sampling distributions mentioned so far are static during the execution of a simulation algorithm, that is, they do not change. The *heuristic importance sampling distribution* [55, 97] is dynamically changing during execution of the simulation algorithm. The basic idea is that the sampling distribution is updated depending on the sample trials generated: every time after k sample trials have been generated, the joint probability distribution represented by the Bayesian belief network is estimated, and a weighted sum of this estimated distribution and the previously used sampling distribution is taken as the new sampling distribution. We will not consider this sampling distribution in the sequel.

5.1.3 Sample Trial Generator

The second component of a simulation algorithm for Bayesian belief networks is the sample trial generator. This generator assigns a value to each variable or set of variables of a network according to a sampling distribution.

In the simulation scheme known as *equiprobable sampling* a sample trial generator is used in which nodes are assigned a value in random order. For the generated configuration x_V of the set of all variables V the trial score is

$$p = \frac{Pr_B(V = x_V)}{Pr_S(V = x_V)} = \frac{\prod_{u \in V} \gamma_u(u = x_u | \pi_u = x_{\pi_u})}{\prod_{u \in V} r_u}.$$

Since in this fraction the denominator is the same for all configurations, it will have no influence on the normalization step at the end of a simulation algorithm. Therefore, it can be omitted and we take the trial score

$$p = \prod_{u \in V} \gamma_u(u = x_u | \pi_u = x_{\pi_u}).$$

When a sample trial generator is used in which all nodes are sampled according to a forward sampling distribution, we speak of *logic sampling* [55], *evidence weighting* [37, 97], or *likelihood weighting* depending on the scoring method; in this thesis, we use the latter phrase. Before a node u can be sampled, its parents must have been assigned a value first. To accomplish this, likelihood weighting is performed by first assigning values to the root nodes (if they are not all evidence nodes), and then assigning values to nodes of which all parents have been assigned a value, until all nodes have been assigned a value. So, the order in which the nodes are assigned a value is a topological ordering.

For a configuration x_V thus generated, we have that the trial score is

$$p = \frac{Pr_B(V = x_V)}{Pr_S(V = x_V)} = \frac{\prod_{u \in V} \gamma_u(u = x_u | \pi_u = x_{\pi_u})}{\prod_{u \in V} \gamma_u(u = x_u | \pi_u = x_{\pi_u})} = 1.$$

If some evidence is entered into the network, a sample generator can be used in which nodes are interchangeably sampled according to the forward and backward sampling distributions. The order in which the nodes are sampled has to meet the following two requirements:

1. a set of nodes can only be backward sampled if it is the parent set π_u of an instantiated node u.
2. a node can only be forward sampled if all its parents are instantiated.

Any order that satisfies these two requirements can be used to generate a sample trial. Note that such an order not necessarily is a topological ordering on the nodes. The influence of the ordering on the convergence of the approximated probabilities to the exact probabilities is still an open question; however, it has been suggested [38] that it may be beneficial to backward sample evidence nodes with assessment functions that have values close to zero since this would generate sample trials with a larger trial score and hence result in more representative sample trials.

For a configuration x_V thus generated, the trail score is computed as follows. Let A be the set of nodes that have been backward sampled and let F be the set of nodes to which forward sampling has been applied. Then, the trial score is

$$p = \frac{Pr_B(V = x_V)}{Pr_S(V = x_V)} = \frac{\prod_{u \in V} \gamma_u(u = x_u | \pi_u = x_{\pi_u})}{\prod_{u \in F} \gamma_u(u = x_u | \pi_u = x_{\pi_u}) \prod_{u \in A} \gamma_u(u = x_u | \pi_u = x_{\pi_u}) / \alpha_u}$$

By dividing out common terms, we find,

$$p = \prod_{u \in V \setminus FA} Pr(u = x_u | \pi_u = x_{\pi_u}) \cdot \prod_{u \in A} \alpha_u.$$

When a sample trial generator is used in which all nodes are assigned a value according to a Markov blanket sampling distribution, we speak of *Gibbs sampling* or *Pearl's scheme*. For this scheme, an initial configuration is necessary, because the Markov sampling distribution depends on such a configuration. The initial configuration is generated by using one of the other simulation methods. From this initial configuration, a new configuration is generated by assigning a value to a single node u with a probability given by the Markov blanket sampling distribution of u with the configuration of the variables in the network. Then the score for u can be updated. This process is applied to every node in $V \setminus E$. The thus generated configuration is the sample trial generated by Markov blanket sampling. For assigning values to the nodes, there is no restriction on the order in which the nodes are dealt with. The order may influence the choice of the configuration though.

Pearl [80] showed that with a trial score $p = 1$ for each sample trial, sample scores tend to converge to the actual probabilities. Like the forward and backward sampling distributions, also this distribution results in good samples, as long as the assessment functions do not contain values close to zero.

5.1.4 Scoring Methods

The third component of a simulation algorithm for Bayesian belief networks is the scoring method. This scoring method is used to calculate the actual approximation of probabilities by assigning a weight to each configuration generated. For each variable u in the network and for every value in Ω_u, a counter is used for calculating the sample scores. For every sample trial in a sample, some or all sample scores are updated using the trial score of this sample. We describe two scoring methods, simple scoring and Markov blanket scoring.

When the *simple scoring* method is used, the trial score p of a sample trial x_V is added to the sample score of each variable's value appearing in x_V. Note that not all scores are updated but only the scores of the values in x_V. Consider once more the Bayesian belief network shown in Figure 5.2. Consider the configuration $a = 0$, $b = 0$, $c = 1$ with the trial score 0.072 obtained by equiprobable sampling. When using the simple scoring method, the sample scores for $a = 0$, $b = 0$ and $c = 1$ are increased by 0.072.

With *Markov blanket scoring* [80], the trial score p is weighted before it is added to the sample score. Instead of adding p to the sample score of a value x_u of node u appearing in the sample trial x_V, the value

$$p \cdot Pr_S(u = x_u | M_u = x_{M_u})$$

is added, where Pr_S is the Markov blanket sampling distribution. Consider once more the Bayesian belief network shown in Figure 5.2 and consider the configuration $a = 0$, $b = 0$, $c = 1$ with $p = 0.072$. For node b, we have $Pr_S(b = 0 | a = 0, c = 1) = 9/17$ and $Pr_S(b = 1 | a = 0, c = 1) = 8/17$. The sample score for $b = 0$ now is increased with $0.072 \cdot 9/17$ and the sample score for $b = 1$ is increased with $0.072 \cdot 8/17$. The sample scores for $a = 0$, $a = 1$, $c = 0$, $c = 1$, and $c = 2$ are updated in a similar fashion.

Note that for the equiprobable and likelihood weighting schemes, additional work needs to be performed in computing the probabilities for the Markov blankets. When used with Markov blanket sampling, however, these probabilities are already available, as they have been computed for generating the configuration.

5.2 A Stratified Simulation Scheme

In this section, we will present a new sample trial generator. It is based on a popular statistical technique known as stratification [85] that aims at obtaining sample trials that are evenly distributed over the sample space. The basic idea of stratification is to divide the sample space into so-called strata and then choose a fixed number of sample trials in each stratum. In this way, it is not possible that no sample trials are taken from a large area of the sample space. As a result, the stratified simulation scheme tends to yield samples from which better approximations are derived.

First, we will describe how to apply stratification to inference in Bayesian belief networks in theory. Then, we show how stratification can be implemented efficiently. To conclude, we consider some optimizations and variations of the stratified scheme.

5.2.1 Stratification for Bayesian Belief Networks

To apply stratification to inference in Bayesian belief networks, we first assign an ordering on the configurations of the variables in the network, and associate a unique subinterval of $[0..1]$ to each configuration. The unit interval is our sample space. There is a large freedom in selecting strata. In our approach, we will split the sample space $[0..1]$ into m equally likely strata and choose one sample trial from each stratum.

Let the variables in the set V be ordered $v_1 < \ldots < v_n$, $n \geq 1$. Let Pr_S be a sampling distribution on V. The elements of the outcome space Ω_i of v_i, $i = 1, \ldots, n$, will be denoted by the integers $0, 1, \ldots, r_i \Leftrightarrow 1$, where r_i is the number of values v_i may adopt. The configurations of V are now taken to be ordered according to the ordering on the integers taking the order of the variables in account, that is, $x_V < x'_V$ if there is an $i \in \{1, \ldots, n\}$ such that $x_{v_j} = x'_{v_j}$ for $j = 1, \ldots, i \Leftrightarrow 1$ and $x_{v_i} < x'_{v_i}$ where x_{v_k} and x'_{v_k} conform to x_V and x'_V respectively. For ease of exposition, assume that there are no variables observed, that is, $E = \emptyset$. With each configuration x_V of V, we associate an interval $I(x_V)$ defined by,

$$I(x_V) = [lo(x_V), hi(x_V)),$$

where

$$lo(x_V) = \sum_{x'_V < x_V} Pr_S(V = x'_V),$$

and

$$hi(x_V) = \sum_{x'_V \leq x_V} Pr_S(V = x'_V).$$

Configuration	Probability	Accumulated probability	Associated interval
000	0.072	0.072	[0.000,0.072)
001	0.072	0.144	[0.072,0.144)
002	0.096	0.240	[0.144,0.240)
010	0.048	0.288	[0.240,0.288)
011	0.048	0.336	[0.288,0.336)
012	0.064	0.400	[0.336,0.400)
100	0.072	0.472	[0.400,0.472)
101	0.096	0.568	[0.472,0.568)
102	0.072	0.640	[0.568,0.640)
110	0.108	0.748	[0.640,0.748)
111	0.144	0.892	[0.748,0.892)
112	0.108	1.000	[0.892,1.000)

Table 5.1: Ordered configurations, associated probabilities and intervals.

So, the lower bound of an interval $I(x_V)$ is the probability that V takes a configuration smaller than x_V and the size of the interval $I(x_V)$ equals the probability of selecting x_V according to Pr_S. In this way, the unit interval $[0..1]$ is divided into subintervals such that for every number $f \in [0..1]$ there is a unique configuration x_V of V such that $f \in I(x_V)$. Consider once more the Bayesian belief network from Figure 5.2, where $V = \{a, b, c\}$. Suppose that the variables are ordered $a < b < c$. Let Pr_S be the joint probability distribution on V defined by the network. For the configuration $a = 0$, $b = 1$, $c = 0$, that is, the configuration $V = 010$, we have that the interval $I(010)$ equals $[0.24, 0.288)$, since $lo(010) = Pr(V = 000) + Pr(V = 001) + Pr(V = 002) = 0.24$ and $hi(010) = lo(010) + Pr(V = x_V) = 0.24 + 0.4 \times 0.4 \times 0.3 = 0.288$. Table 5.1 lists the intervals for the various configurations of V.

The stratified simulation scheme now uses the intervals $I(x_V)$ of the configurations x_V of V to select sample trials. Basically, it chooses a number f randomly from the unit interval and then yields the configuration x_V of V such that $f \in I(x_V)$. In our example, suppose that the number $f = 0.246$ is chosen. Since f is in the interval $I(x_V) = [0.24, 0.288)$ of the configuration $x_V = 010$, the sample trial $a = 0$, $b = 1$, $c = 0$ is yielded.

The concept of intervals associated to configurations of V can be generalized to intervals associated to prefixes of configurations of V. We define the prefix of k bits of configuration x_V, $1 \leq k \leq n$, denoted as $pref_k(x_V)$, by $x_{v_1} \ldots x_{v_k}$. For example, $pref_3(0111) = 011$ and $pref_1(0111) = 0$. Then, the intervals generalized to prefixes $I_k(x_V)$ associated with configuration x_V is defined for $1 \leq k \leq n$ by

$$I_k(x_V) = [lo_k(x_V), hi_k(x_V)),$$

where

$$lo_k(x_V) = \sum_{pref_k(x'_V) < pref_k(x_V)} Pr_S(V = x'_V),$$

and

$$hi_k(x_V) = \sum_{pref_k(x'_V) \leq pref_k(x_V)} Pr_S(V = x'_V),$$

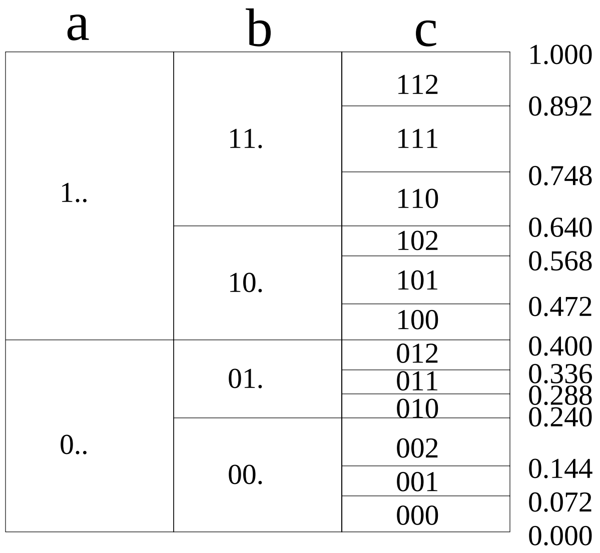


Figure 5.3: Intervals of prefixes.

and for $k = 0$,

$$I_k(x_V) = [0, 1).$$

So, the lower bound of an interval $I_k(x_V)$ is the probability that the variables $v_1 \dots v_k$ take a configuration smaller than $pref_k(x_V)$ and the size of the interval $I_k(x_V)$ equals the probability of selecting $v_1 \dots v_k$ conform to x_V according to Pr_S. Note that for $k = n$ we have $I_k(x_V) = I(x_V)$. Also note that $I_k(x_V) \subseteq I_{k-1}(x_V)$ for all $k = 1, \dots, n$. Consider once more the Bayesian belief network of Figure 5.2. Figure 5.3 shows the various intervals of all possible configurations. For example, $I_2(01.)$, where for the . in 01. any value of c may be substituted, starts at 0.24 since $Pr_S(pref_2(x_V) < 01) = Pr_S(V = 000) + Pr_S(V = 001) + Pr_S(V = 002) = 0.24$ and ends at 0.4 since $Pr_S(pref_2(x_V) = 01) = Pr_S(V = 010) + Pr_S(V = 011) + Pr_S(V = 012) = 0.16$.

5.2.2 An Algorithm for the Stratified Simulation Scheme

Based on the concept of stratification outlined above, we now formulate a simulation scheme. The basic idea is as follows. We divide the unit interval $[0..1]$ into m equal disjoint parts called *strata*, where m is the number of required sample trials. For each stratum we generate one random number f. These m random numbers are chosen in the unit interval and these numbers are considered in ascending order. Now suppose that the numbers $f_1 = 0.246$, $f_2 = 0.399$, and $f_3 = 0.6789$ have been generated. The sample trial corresponding to the first number is $x_{V,1} = 010$, to the second $x_{V,2} = 012$, and to the third $x_{V,3} = 110$. Observe that for the samples $x_{V,1}$ and $x_{V,2}$ only the least significant number has changed. In general, when the random numbers are considered in ascending order, then only the k least significant bits change and the $n \Leftrightarrow k$ most significant bits do not. This property can be exploited to get an efficient simulation scheme by dynamically changing values of these least significant variables. When we are looking for an interval that contains f and the previous sample trial is configuration $x_{V,i-1}$, we first check if f is in $I_n(x_{V,i-1})$. If it is not, we check if it is in $I_{n-1}(x_{V,i-1})$ and so forth, until we find a k such that f

```
$l_0 \leftarrow 0$;  $h_0 \leftarrow 1$
for  $i \leftarrow 1$  to  $n$  do
    $l_i \leftarrow 0$
    if $v_i \in E$ then
        $x_{v_i} \leftarrow e_i$
        $h_i \leftarrow h_{i-1}$
    else
        $x_{v_i} \leftarrow 0$
        $h_i \leftarrow h_{i-1} * Pr_S(v_i = 0 | V_i = x_{V_i})$
```

Figure 5.4: Initialization of the stratified scheme.

is in $I_k(x_{V,i-1}) = [lo_k(x_{V,i-1}), hi_k(x_{V,i-1}))$. Now observe that for all j, $lo_j(x_{V,i-1})$ is smaller than f. So, only $hi_j(x_{V,i-1})$ need to be considered; looking for k such that $hi_k(x_{V,i-1}) > f$ and $hi_{k+1} < f$ is sufficient. Since $hi_k(x_{V,i-1})$ is a descending function of k, this procedure can be performed with binary search.

In Figure 5.4 pseudo-code for the initialization method for the stratified scheme is shown and in Figure 5.5 pseudo-code for generating the ith sample trial is shown. The values of the variables in a sample trial in the making are stored in the array x_V; we keep track of the various prefix intervals of the configuration stored in x_V in the arrays l and h for respectively the lower and upper bounds.

During initialization, the lowest ordered configuration and the associated intervals are determined. So, a configuration x_V^0 is generated in which each variable is assigned a value 0 except when there is evidence for the variable. Evidence variables are assigned their observed values. Note that evidence nodes do not contribute to the interval. Since we consider the lowest ordered configuration, the lower bounds of all prefix intervals associated with x_V^0 are 0, that is, $lo_j(x_V^0) = 0$, $j = 0, \ldots, n$. The upper bound $hi_j(x_V^0)$ equals by definition $\sum_{pref_j(x'_V) \leq pref_j(x_V^0)} Pr_S(V = x'_V)$ which for the initial configuration x_V^0 reduces to $Pr_S(v_1 = x_1^0, \ldots, v_j = x_j^0)$ where v_j is the jth variable according to the ordering on V. By the definition of conditional probability, we have $Pr_S(v_1 = x_1^0, \ldots, v_j = x_j^0) = Pr_S(v_j = x_j^0 | v_1 = x_1^0, \ldots, v_{j-1} = x_{j-1}^0) \cdot Pr_S(v_1 = x_1^0, \ldots, v_{j-1} = x_{j-1}^0)$. And, since $Pr_S(v_1 = x_1^0, \ldots, v_{j-1} = x_{j-1}^0)$ equals $hi_{j-1}(x_V^0)$, and we write V_j for $V_j = \{v_i | 1 \leq i < j\}$, we can calculate $hi_j(x_V^0)$ by $Pr_S(v_j = x_j^0 | V_j = x_{V_j}^0) \cdot hi_{j-1}(x_V^0)$.

Figure 5.5 shows pseudo-code for generating the ith sample trial. Basically, a random number f is generated, and the first variable v_j in the ordering is identified that needs to be assigned a new value. Then, one by one each variable $v_j, \ldots, v_n$ is assigned a value and its associated intervals are determined.

First, a number f from the ith stratum $[(i \Leftrightarrow 1)/m, i/m)$ is selected randomly. Using binary search, the first variable v_j for which $h_j < f$ and $h_{j-1} > f$ is identified. This is the first variable that need to be assigned a new value. All lower ordered variables remain assigned the same value as in the $i \Leftrightarrow 1$th configuration as we argued above. For the variables v_j up to v_n a new value will be calculated in ascending order. To assign a value to v_j, the configuration x_V and its associated interval

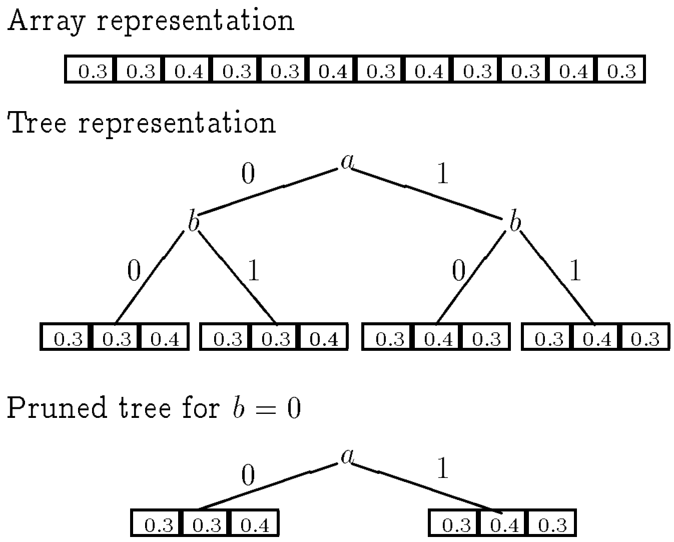
```
$f \leftarrow (\text{random}[0:1) + i \Leftrightarrow 1)/m$
$j \leftarrow$ Binsearch $(f, h)$
while $j <= n$ do
    if $v \in E$ then
        $l_j \leftarrow l_{j-1}$
        $h_j \leftarrow h_{j-1}$
    else
        $k \leftarrow 0$
        $l_j \leftarrow l_{j-1}$
        $h_j \leftarrow l_j + (h_{j-1} \Leftrightarrow l_{j-1}) * Pr_S(v_j = k|V_j = x_{V_j})$
        while $f > h_j$ do
            $k \leftarrow k + 1$
            $l_j \leftarrow h_j$
            $h_j \leftarrow l_j + (h_{j-1} \Leftrightarrow l_{j-1}) * Pr_S(v_j = k|V_j = x_{V_j})$
        $x_{v_j} \leftarrow k$
    $j \leftarrow j + 1$
return$(x_V)$
```

Figure 5.5: The sample trial generator of the stratified simulation scheme.

$I_j(x_V) = [lo_j(x_V), hi_j(x_V))$ needs to be determined that includes f. To this aim, we step through the state space of v_j by taking the values in ascending order until we find a value of v_j such that $lo_j(x_V) \leq f < hi_j(x_V)$.

The lower and upper bounds of the various prefix intervals to be considered for v_j are calculated from the bounds for v_{j-1} as follows. If v_j is an evidence node, then the boundaries are the same as for v_{j-1}. If v_j is not an evidence node, then $lo_j(x_V)$ and $hi_j(x_V)$ are bounded by the boundaries of v_{j-1}. From the definition of intervals over prefixes and the definition of conditional probability it follows that when $v_j = 0$ in x_V then $lo_j(x_V) = lo_{j-1}(x_V)$ and $hi_j(x_V) = lo_j(x_V) + (hi_{j-1}(x_V) \Leftrightarrow lo_{j-1}(x_V)) \cdot Pr_S(v_j = 0|V_j = x_{V_j})$. Further, when x'_V is the same configuration as x_V where the value of v_j is incremented by one, we have that $lo_j(x'_V) = hi_j(x_V)$ and $hi_j(x'_V) = lo_j(x'_V) + (hi_{j-1}(x_V) \Leftrightarrow lo_{j-1}(x_V)) \cdot Pr_S(v_j = k|V_j = x_{V_j})$. So, the bounds of the intervals can be efficiently calculated.

The stratified simulation scheme is a sample trial generator. There are several ways of defining the sampling distribution Pr_S. When Pr_S is chosen such that all variables are independent and $Pr_S(v_j = x_j) = 1/r_j$, where r_j is the number of values v_j can take, all configurations are equiprobable and this scheme will be referred to as the *stratified equiprobable scheme*. In this case, $Pr_S(v_j = x_j|V_j = x_{V_j}) = 1/r_j$ for all $x_j \in \Omega_j$ and $x_{V_j} \in \Omega_{V_j}$. For Pr_S it is also possible to take the distribution represented by the Bayesian belief network. In this case when the ordering on the variables is a topological ordering $Pr_S(v_j = x_j|V_j = x_{V_j}) = \gamma_j(v_j = x_j|\pi_{v_j} = x_{\pi_{v_j}})$ for all $x_j \in \Omega_j$ and $x_{V_j} \in \Omega_{V_j}$. This scheme will be referred to as the *stratified likelihood scheme*. The trial scores for the stratified equiprobable scheme and the stratified likelihood scheme are the same as for the equiprobable scheme and the likelihood


Array representation

Tree representation

Pruned tree for $b = 0$

Figure 5.6: Data structures for storing assessment functions.

scheme respectively.

5.2.3 Optimizations of the Simulation Scheme

In this section, we briefly point out some optimizations of the stratified simulation scheme.

It is desirable to generate many representative sample trials in a small amount of time to get a good approximation. Since in generating sample trials the assessment functions of the Bayesian belief network at hand are used, it is important to carefully design the data-structure for storing these assessment functions. Recall that the assessment functions may be looked upon as tables of conditional probabilities. Such a table can be stored in an array. For example, in CABeN [25], a collection of algorithms for belief networks, probability tables are stored this way. The basic idea of array-storage is illustrated in Figure 5.6: in the depicted array, the probabilities for the variable c from the example of Figure 5.2 are stored. A disadvantage of storing an assessment function in an array is that for finding a specific probability, an index needs to be calculated. In general, the calculation of such an index is computationally expensive. For the example of Figure 5.6, the probability that c takes value x_c given that a and b take value x_a and x_b respectively is stored at index $x_c + r_c \cdot x_b + r_c \cdot r_b \cdot x_a$.

A search tree offers an alternative data structure for storing assessment functions. The search tree storing the assessment function for a node u is a tree in which the leafs contain the values of the function γ_u, the internal nodes are associated with variables in the parent set of u, and branches outgoing from a node associated with a variable v are labeled with the values that v may take. The basic idea of storing an assessment function in a search tree is shown in Figure 5.6 where the search tree for the assessment function of the variable c from the example of Figure 5.2 is depicted. To determine the value of an assessment function for a given configuration x_{π_u} of π_u, the tree is searched by starting at the root. On every node associated with a variable v the branch labeled with the value x_v conform to x_{π_u} is taken to arrive at a new node or a leaf. This process is repeated until a leaf is reached. So, for finding a specific probability, a pointer may be passed through the tree and no multiplications need to be performed.

The search tree offers another advantage when the following technique is used. When evidence is observed, outgoing arcs of the observed nodes can be removed and the assessment functions updated to arrive at a new Bayesian belief network that represents the conditional joint probability distribution given the observed evidence [39]. One of the motivations of applying the technique is that if variable u is observed to be x_u then inference algorithms will not use probabilities of children of u conditioned on configurations not involving x_u. Since these probabilities will not be accessed anymore, they can be removed from the search tree representing the assessment function. When a search tree is used for storing the assessment function, the tree is pruned by replacing a node associated with an observed variable by the subtree attached to the branch labeled with its observed value. Consider once more Figure 5.6. When b has been observed to have the value 0, the search tree for the assessment function of node c can be pruned arriving at the tree depicted in the lower part of Figure 5.6. Pruning a search tree is an almost trivial operation. Pruning an array on the other hand would require considerable computational effort.

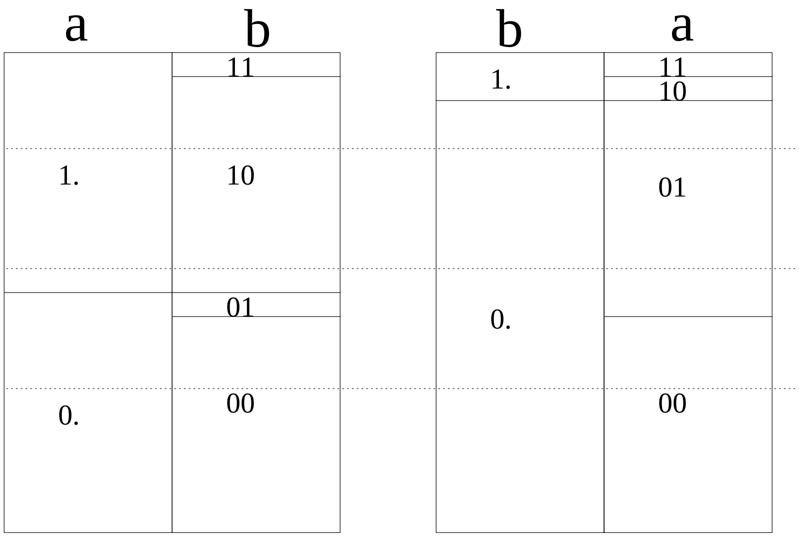


Figure 5.7: Influence of order on intervals.

Not only the choice of the data structure for storing the assessment functions is important for optimal computational performance. Recall that the stratified likelihood scheme builds on a topological ordering on the variables. In general, a directed acyclic graph allows several topological orderings of its variables. To fully exploit the reduction in time achieved by the stratified schemes compared to ordinary schemes, variables with high probabilities in their assessment functions should occur foremost in the ordering; in that case large intervals will occur for small prefixes and therefore the lower ordered variables won't need a change of value too often. Figure 5.7 illustrates the intervals for two independent binary variables a and b with probabilities 0.5 and 0.9 of being 0 respectively. When a is ordered first, we have

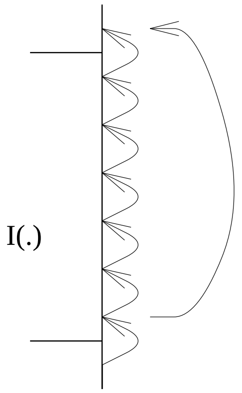


Figure 5.8: Skipping steps for large intervals.

the intervals shown on the left-hand side of Figure 5.7 and when b is ordered first, we have the intervals shown on the right-hand side. When three sample trials are required and the values 0.3, 0.55 and 0.8 have been selected, we see that when a is ordered first, four value assignments take place and when b is ordered first, only three value assignments take place.

So, when choosing a topological ordering of the variables, the assessment functions should be taken in consideration. An example of a criterion for choosing between various orderings is to select the ordering in which the nodes are ordered first with a maximal value of $\sum_{u\pi_u \in \Omega_{u\pi_u}} Pr(u = x_u|\pi_u = x_{\pi_u})^4/r_{u\pi_u}$ which assigns extra weight to probabilities close to one, whereas small probabilities do not contribute much to the sum.

So far, we assumed that a random number in each stratum was chosen. However, also the median of the stratum can be taken. Then, for the ith sample trial, $i = 1, \ldots, m$, the value of f_i is taken to be

$$f_i = \frac{i \Leftrightarrow 0.5}{m}$$

instead. At least for the lower ordered nodes, no large changes in approximations are expected because the values assigned to these nodes will be the same as when random values are selected. In fact, these estimates will tend to become slightly better because fewer errors due to random fluctuations are introduced. For variables high in the ordering, selecting the median of a stratum has the same effect as choosing a random number in the stratum.

In case the values of f_i are taken to be the median of a stratum, we know on fore-hand which values f_i will be visited. Now observe that for a configuration x_V that has a large interval $I(x_V)$, the generated sample trials may not change for many successive numbers; this idea is illustrated in Figure 5.8. Yet, every time that a new sample trial is generated for a new number, a binary search is performed. This work can be saved, by simply passing over the calculations for the numbers for which on fore-hand it is known that they result in the same sample trial. These numbers can be identified from the boundaries of the interval is stored in h_n and l_n. Note that

```
Initialize
$i \leftarrow 1$
while $i \leq m$ do
    $x_{V\backslash E} \leftarrow$ generate configuration $i$
    $p \leftarrow Pr_B(V\backslash E = x_{V\backslash E}, E = x_E)/Pr_S(V\backslash E = x_{V\backslash E}, E = x_E)$
    $\delta = \lfloor \frac{h_n - f_i}{m} \rfloor + 1$
    update scores $\delta \cdot p$
    $i \leftarrow i + \delta$
Normalize scores
```

Figure 5.9: Simulation framework for the modified scheme.

$$\delta = \left\lfloor \frac{hi(x_V) - f_i}{m} \right\rfloor + 1 \tag{5.8}$$

numbers can be passed over, where $\lfloor . \rfloor$ denotes the integer part. In the general framework shown in Figure 5.1, we increment the counter i of the sequence f_i with δ instead of one. In this way, we save the work of generating the same configuration over and over again for different successive values of f_i. Observe that in the basic scheme, the configuration is scored δ times, and in the new scheme the configuration is scored only once. So, the trial score p has to be adapted to compensate for skipping configurations in the modified scheme. This is achieved by multiplying p by δ. We refer to the new scheme in which values of f_i are passed over as the *modified stratification scheme* in contrast to the *standard* stratified scheme. Figure 5.9 shows the modified stratification scheme in pseudo-code.

Care must taken when networks with many variables are used; the values of $lo_k(x_V)$ and $hi_k(x_V)$ may erroneously be calculated as equal due to numerical round off errors. Therefore, the representation size used for $lo_k(x_V)$ and $hi_k(x_V)$ need to be taken large enough.

5.3 Performance Analysis

Usually, the likelihood weighting scheme is the simulation scheme that gives the best performance when both computation time and error properties are considered. In this section we compare the computational complexity of the stratified simulation schemes with the likelihood weighting simulation scheme. In Section 5.3.1 we compare the standard stratified scheme with the likelihood weighting scheme, and in Section 5.3.2 we compare the standard stratified scheme with the modified stratified scheme.

5.3.1 Performance of the Stratified Scheme

The basic idea of using the concept of stratification as described in Section 5.2 is that in generating a new sample trial, the stratified scheme saves the work of determining values for several variables by using the values from the previous sample trial. To determine which variables require a new value, a binary search is performed which costs at most $\log n$ comparisons where n is the number of variables considered. Based on these ideas, we can determine the computational savings of the stratified likelihood scheme compared to the likelihood weighting scheme.

5.1 **Theorem** Let V be a set of n binary variables. Let Pr_S be a sampling distribution over V and let m be the number of trials of the sample to be generated. Let x and y be the number of value assignments in generating the sample with Pr_S by the likelihood weighting and the stratified likelihood weighting simulation scheme, respectively. Let α be the relative cost of a comparison with respect to a value assignment. Then,

$$x - y > m \cdot (\lfloor \log m \rfloor - \alpha \cdot \log n).$$

□

Proof: We begin by determining the number x of value assignments performed by the likelihood weighting scheme. In this scheme, in each sample trial that is generated all variables are considered and assigned a value. So, $x = m \cdot n \cdot \beta$ value assignments are performed.

Now, consider the number y of value assignments performed by the stratified likelihood weighting scheme. We observe that the most significant variable gets assigned a value at most twice, the second most significant non-evidence variable at most four times, etcetera. The $\lfloor \log m \rfloor$th variable gets assigned a value at most $2^{\lfloor \log m \rfloor}$ times. We observe that every variable gets assigned a value at most m times. Therefore, the ith variable, $i > \lfloor \log m \rfloor$, gets assigned a value at most m times. So, at most

$$\sum_{i=1}^{\lfloor \log m \rfloor} 2^i + (n - \lfloor \log m \rfloor) \cdot m$$

variable assignments are performed. Note that the amount of work involved in assigning a value to a variable is the same as for the likelihood weighting scheme. In the stratified likelihood weighting scheme in addition at most m times a binary search is performed each taking a computational effort of $\alpha \cdot \log n$. The binary searches take a computational effort of $\alpha \cdot m \cdot \log n$. Using $\sum_{k=0}^{x} 2^k = 2^{x+1} - 1$, we find that

$$y < 2^{\lfloor \log m \rfloor + 1} - 1 + (n - \lfloor \log m \rfloor) \cdot m + \alpha \cdot m \cdot \log n.$$

So, for the difference $x - y$ we have

$$x - y > -2^{\lfloor \log m \rfloor + 1} + 1 + \lfloor \log m \rfloor \cdot m - \alpha \cdot m \cdot \log n.$$

Using $-2^{\lfloor \log m \rfloor + 1} \geq -2^{\log m + 1} = -m - 1$, it follows that,

$$x - y > \lfloor \log m \rfloor \cdot m - \alpha \cdot m \cdot \log n,$$

which proves the theorem. □

From the theorem, we have that the likelihood weighting scheme has a worst-case computational complexity of $O(n \cdot m)$ and the stratified likelihood weighting scheme has a computational complexity of $O((n - \log \frac{m}{n}) \cdot m)$. From these observations, we conclude that if the number of sample trials m is larger than the number of variables n, which is almost always the case, the stratified likelihood weighting scheme is more efficient than the likelihood weighting scheme. Similar observations apply if we compare the equiprobable simulation scheme with the stratified equiprobable scheme.

In general, probability estimates become more accurate as the number of sample trials increases. Dagum and Horvitz [28] showed that for the likelihood weighting scheme, to output a probability of a value of a variable u being x_u $Pr(u = x_u|E = x_E)$ with probability higher than $1 - \delta$ has relative error smaller than ϵ, at least

$$a \cdot \log(4/\delta)/(\epsilon^2 Pr(u = x_u|E = x_E))$$

sample trials are required where a is the maximum value of the sampling distribution. Consider once more the example of Figure 5.3. For $m > 5$ sample trials, always $\lfloor 0.4 \cdot m \rfloor$ trails with $a = 0$ and $\lfloor 0.6 \cdot m \rfloor$ trails with $a = 1$ will be generated. This leaves at most one variable to be assigned a value and the estimate of the probabilities will be very accurate for a. So, the algorithm produces better samples, a point stressed in [17] to be very important. Especially for variables that are low in the ordering good sample trials are produced. We feel that the bound of Dagum and Horvitz may be taken as an upper bound to the number of sample trials to be generated.

5.3.2 Standard Stratified Scheme versus Modified Stratified Scheme

In this section, we give an impression of the amount of work that needs to be performed for the modified stratified scheme compared to the standard stratified scheme. So, we are interested in finding the configurations for which the associated intervals are larger than the size of the strata used since for these configurations the modified stratification scheme will take less computational time than the standard stratified scheme. We will consider the case that all variables in the Bayesian belief network are binary, independent, and identically distributed.

Let V be a set of n binary variables and let the sampling distribution Pr_S be a joint probability distribution on V such that $Pr_S(V = x_V) = \prod_{u \in V} Pr_S(u = x_u)$ with for all $u \in V$, $Pr_S(u = 1) = p$, $p \leq 0.5$. Now, let s and r be the number of configurations generated by the standard stratified scheme and modified stratified scheme, respectively, with sampling distribution Pr_S and stratum size $1/s$. Then we find that

5.9
$$s - r > \sum_{0 \leq k \leq k_{limit}} \left(\begin{array}{c} n \\ k \end{array} \right) \lfloor s \cdot p^k \cdot (1-p)^{n-k} - 1 \rfloor,$$

where

5.10
$$k_{limit} = \frac{-\log 2 + \log s + n \log(1-p)}{\log(1-p) - \log(p)}.$$

Figure 5.10 shows $\log\lfloor r/s \rfloor$ as a function of s for $n = 30$ and different values of p. Two interesting irregularities can be observed in this figure. The sudden jumps are due to the integer part function $\lfloor . \rfloor$ and the increasing segments are due to the fact that for very small decrements of the stratum size no extra saving occurs because no new configurations are attained.

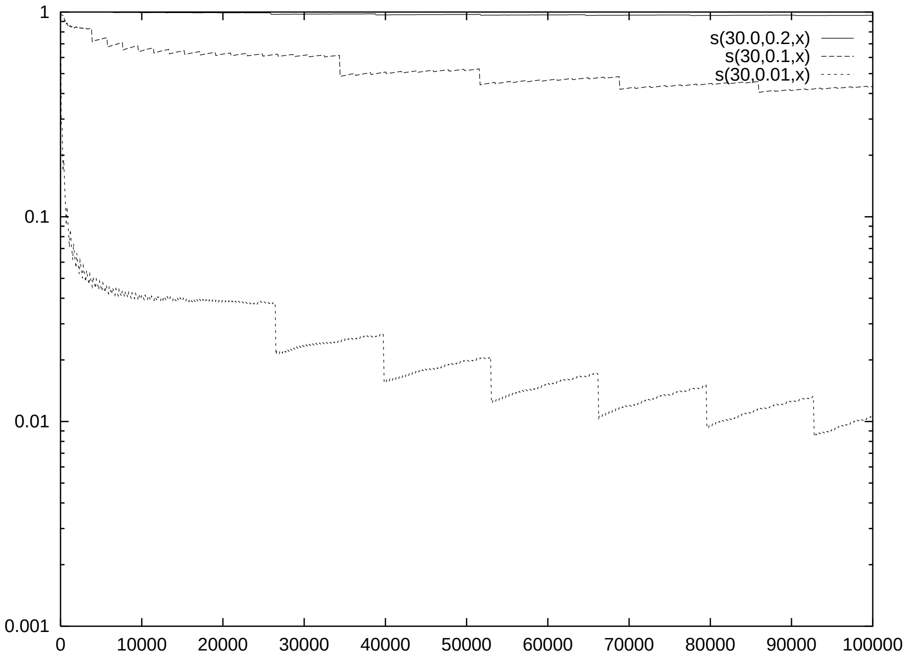


Figure 5.10: $\log\lfloor r/s \rfloor$ as a function of s for $n = 30$ and $p = 0.2$ (upper curve), 0.1 (intermediate) and 0.01 (lower curve).

When the stratum size $1/s$ is less or equal than the minimum of the probabilities of all configurations, that is, less than or equal to p^n, we have, $k_{limit} = n$ and the total saving is

$$\sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) \left(s \cdot p^k \cdot (1-p)^{n-k} - 1 \right) = s - 2^n, \qquad \textbf{5.11}$$

using $\sum_{k=0}^{n} \binom{n}{k} = 2^n$. So, after all 2^n configurations have been generated there is no increment in simulation time of the modified stratified scheme. Note that when $s > 2^n$ there is no reason for simulating because the exact evaluation of the probabilities of all configurations can be done in the same computation time.

We can also compute the savings in terms of the number of operations. Let x and y be the number of value assignments performed by the likelihood weighting and the stratified likelihood weighting simulation scheme, respectively. Let α be the relative cost of a multiplication with respect to a comparison. Then,

$$x - y = ((s - r)\log n + r \cdot \alpha).$$

Asymptotically, the influence of the second term $r \cdot \alpha$ vanishes. The expression shows that the saving increases dramatically with $\log n$ and $1 - p$. This suggests that the modified stratified simulation scheme may be an important improvement over the standard stratification scheme for small values of p under the conditions outlined before under the conditions outlined before.

5.4 Experimental Results

We have performed some experiments to compare the stratified simulation and the modified simulation scheme with likelihood weighting and Pearl's scheme. The aim of these experiments is to get insight in the relative gain in computational performance and approximation accuracy of the stratified schemes compared to the other schemes. Further, we want to get an impression of the influence of the proposed optimizations on the computational performance and approximation accuracy.

In our various experiments, we have used ten randomly generated network structures with twenty-five and ten structures with fifty binary variables. The network structures were generated departing from an arcless graph and an ordering on the variables. First, two nodes u and v are selected randomly, and an arc is added between these nodes: the arc $v \rightarrow u$ is added if $u > v$ and $u \rightarrow v$ otherwise. Then, for each successive arc two nodes are chosen: one node that already has one or more incident arcs, and one that has no incident arc yet. An arc is added between these nodes in the direction that satisfies the ordering. The process is repeated until all nodes have at least one incident arc. Because all arcs are directed from a lower ordered node to a higher ordered node, the resulting graph is guaranteed to be a directed acyclic graph. In fact, this method results in a graph that is a poly-tree. The method generates networks with a bias towards networks with some nodes having a high number of arcs connected as opposed to networks with long strings of nodes. The reason for this bias is that there are more directed acyclic graphs of the former topology in the class over all poly-trees over the specified number of nodes. Realistic networks seem to have the same kind of bias.

For the networks structures thus generated, assessment functions were generated using a random number generator. Since the performance of the stratified schemes depend on the sizes of the intervals, and thus on assessment functions, we considered two distributions for generating the assessment functions. In the first experiment, the random numbers were selected from the unit interval and in the second experiment the number was uniformly selected from $[0..0.1] \cup [0.9..1]$. Each experiment was performed by generating 100 up to 1000 configurations, increasing by 100 in each test and 1000 up to 10000 configurations increasing by 1000 in each test. From each experiment, we have determined the time to execute one of the simulation algorithms and the error in the estimates returned by the algorithm. For measuring the error, we have used the average logarithmic distance of the estimates and the exact probabilities

$$\frac{1}{|V|} \sum_{u \in V} \sum_{x_u \in \Omega_u} Pr(u = x_u) \cdot \log \frac{Pr(u = x_u)}{\hat{Pr}(u = x_u)}$$

where $Pr(u = x_u)$ is the exact probability that the variable u takes value x_u and $\hat{Pr}(u = x_u)$ is probability estimate yielded by simulation. The exact probabilities were calculated using the inference algorithm of Shachter [95]. We have not incorporated evidence in our experiments because evidence only influences the sampling distribution, and not the sample generation. All experiments are performed on a HP-9000 series 700 using a program written in C.

To get an impression of the performance of the stratification technique, we applied the equiprobable scheme, the likelihood weighting scheme, Pearl's scheme, the

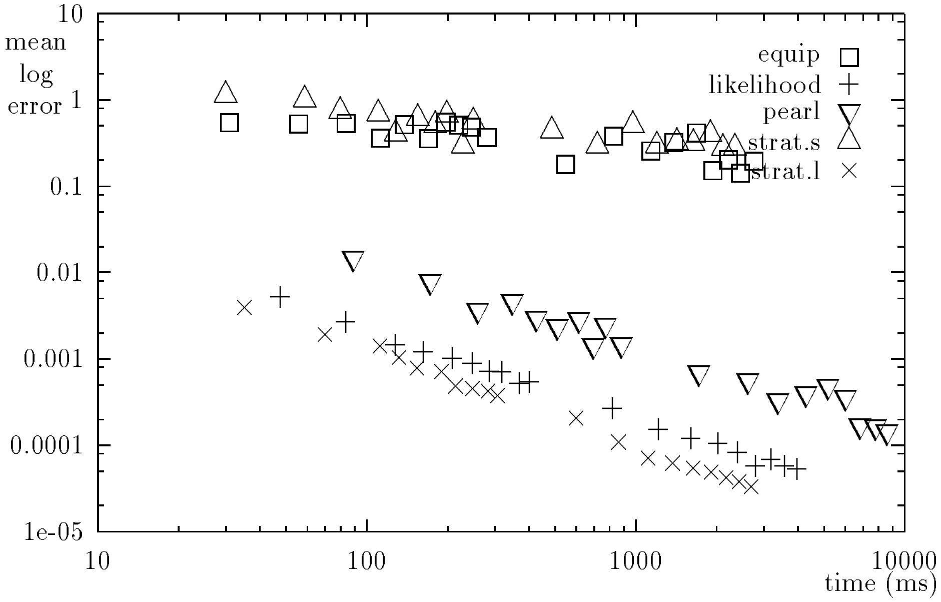


Figure 5.11: Results for different simulation schemes with simple scoring.

stratified equiprobable scheme and the stratified likelihood scheme to the Bayesian belief networks with fifty nodes with simple scoring and with Markov blanket scoring. The ordering of the variables was the same as the order used to generate the networks; the assessment functions were not considered for the ordering. To get an impression of the optimizations, we applied then the stratified likelihood scheme where we have sorted the variables and have taken the median of a stratum. To determine when and how well the modified stratified likelihood scheme performs, this scheme was applied to the Bayesian belief networks with twenty five and fifty nodes with assessment functions generated in both ways described above.

The resulting computation times and errors averaged over the Bayesian belief networks are shown in diagrams in which the average computational time is depicted on the x-axis and the average error in the estimates on the y-axis. The closer the data-points are to the left lower corner of the graph, the better the performance of the scheme.

Figure 5.11 shows the results for the equiprobable scheme (equip), the likelihood weighting scheme (likelihood), Pearl's scheme (pearl), and the stratified schemes for both the equiprobable (strat.s) and likelihood weighting (strat.l) variant. From this figure, it is seen that both the equiprobable scheme and the stratified equiprobable scheme result in very poor approximations. The reason is that most of the sample trials that are generated with these schemes are non-specific for the distribution and many sample trials are required to arrive at a good performance. Though stratification results in a shorter run-time, it does not influence this behavior very much. The figure suggests that with the stratified scheme the error is even larger than with standard equiprobable sampling.

The likelihood weighting scheme performed considerably better both in computation time and error than Pearl's scheme and the equiprobable schemes, as has been reported before [25, 97]. The reason that Pearl's scheme is computationally outperformed by the likelihood scheme is that the latter scheme spends less effort in

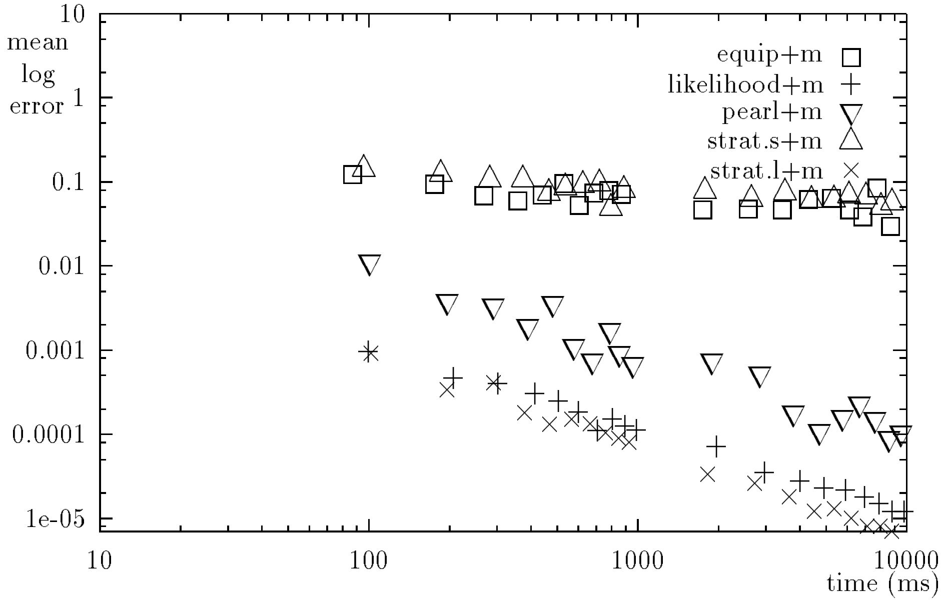


Figure 5.12: Results for different simulation schemes with Markov blanket scoring.

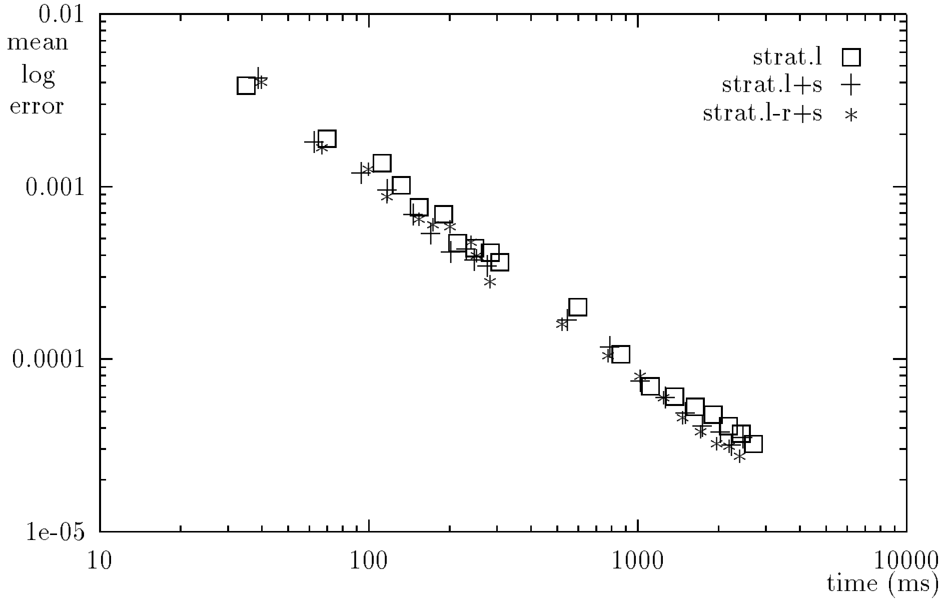


Figure 5.13: Results for various optimizations of the stratified likelihood weighting scheme.

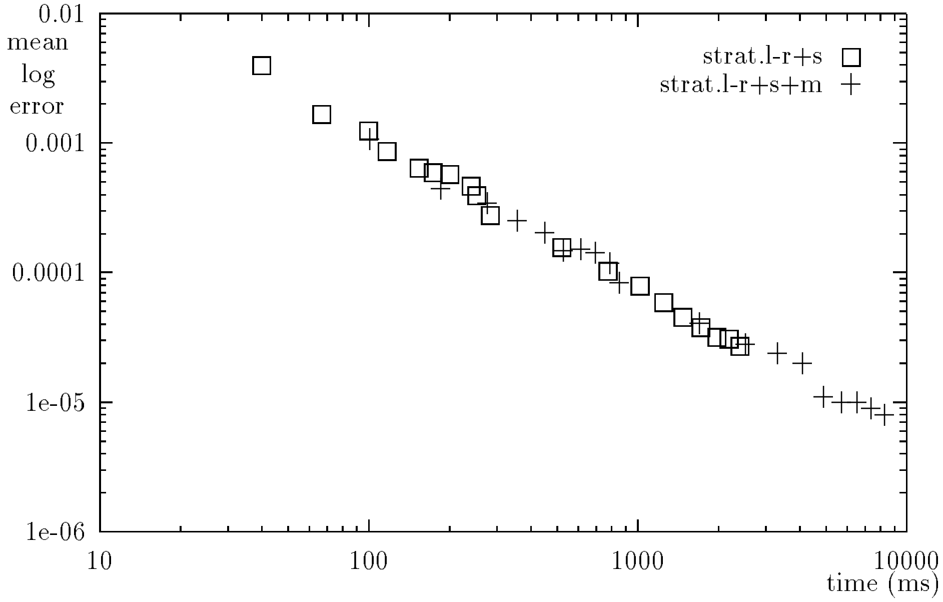


Figure 5.14: Results for the best stratified likelihood weighting scheme with and without Markov blanket scoring.

generating a sample trial. The stratified likelihood weighting scheme has a slightly shorter run-time and a smaller error than the likelihood weighting scheme as expected from the analysis in Section 5.3.1.

Figure 5.12 shows the results for the same schemes as in Figure 5.11, this time using Markov blanket scoring. We note that compared to Figure 5.11, all datapoints have shifted in the direction of the corner right under except for the points of Pearl's scheme, meaning that the estimates become better at the cost of additional computational effort. This could be expected since Markov blanket scoring results in much additional work for all but Pearl's scheme as pointed out in Section 5.1.2. Further, since Markov blanket scoring contributes to every sample score for every sample trial, and not only to those scores for which the values occur in the sample trial, we get a smaller approximation error.

Figure 5.13 shows the effects of incorporating the various optimizations discussed in Section 5.2.3 into the stratified likelihood weighting algorithm (strat.l). We have added sorting the variables to the standard scheme (strat.l+s) and we have modified the standard scheme by taking median of a stratum (strat.l-r+s). The figure suggests that sorting the variables tends to improve the performance of the simulation scheme but only marginally. This could be expected since sorting with the extra criterion influences the order only marginally. The effect of using the median of a stratum instead of a random value is equally marginal. These results were to be expected as the optimizations represent only minor adjustments of the algorithm.

Figure 5.14 shows the results for the best stratified algorithm (that is, with sorting and taking the median of a stratum), with (strat.l-r+s+m) and without (strat.l-r+s) Markov blanket scoring. The figure suggests that the use of Markov blanket scoring tends to improve the probability estimates yet takes extra time because of the additional computational effort that is required. These effects cancel each other out, that is, if the extra computation time necessary for Markov blanket scoring is

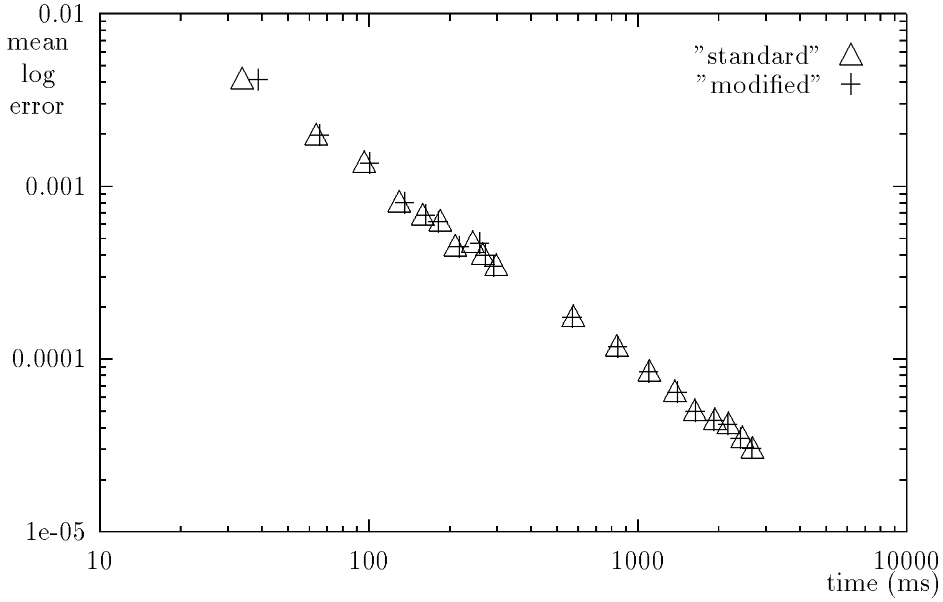


Figure 5.15: Results for the standard and modified stratified scheme.

used for generating extra sample trials with simple scoring, the same error in the estimates can be expected. Therefore, Markov blanket scoring does not seem to help but it also does no harm.

Figure 5.15 shows the results for the standard stratified scheme and the modified stratification scheme. From the figure, it is seen that the resulting error is the same for both schemes. Also, there is hardly any difference in execution time between the two schemes. Note that we would expect a small improvement of the modified stratified scheme over the standard one. The number of generated sample trials for the standard scheme is almost the same as the number of generated sample trials for the modified scheme. As a result, the computational saving obtained by the modified stratification scheme is compensated by the computational effort spent on calculating the number of sample trials to pass over.

Figure 5.16 shows the results for both the standard stratified scheme (hstandard) and the modified stratified scheme (hmodified) for the Bayesian belief networks where the probabilities of the assessment functions have been selected uniformly from the interval $[0..0,1] \cup [0.9..1]$. We observe that even for large numbers of sample trials, only a small computational saving is obtained. So, the number of configurations in which skipping occurs is still very small. Note that both the error and the execution times shown in Figure 5.16 are considerably smaller than the error and execution times shown in Figure 5.15. The reason is that intervals for prefixes of configurations are larger for extreme probabilities. Therefore, fewer transitions of the lower ordered variables occur.

The resulting errors and computation times for Bayesian belief networks with twenty-five nodes are depicted in Figure 5.17 and Figure 5.18 for assessment functions chosen uniformly from $[0..1]$ and $[0..0.1] \cup [0.9..1]$ respectively. In both figures, the modified stratified scheme performs better than the standard stratified scheme. So, the modified stratified scheme can take advantage of the larger size of intervals compared to the fifty node networks. Especially for networks with extreme proba-

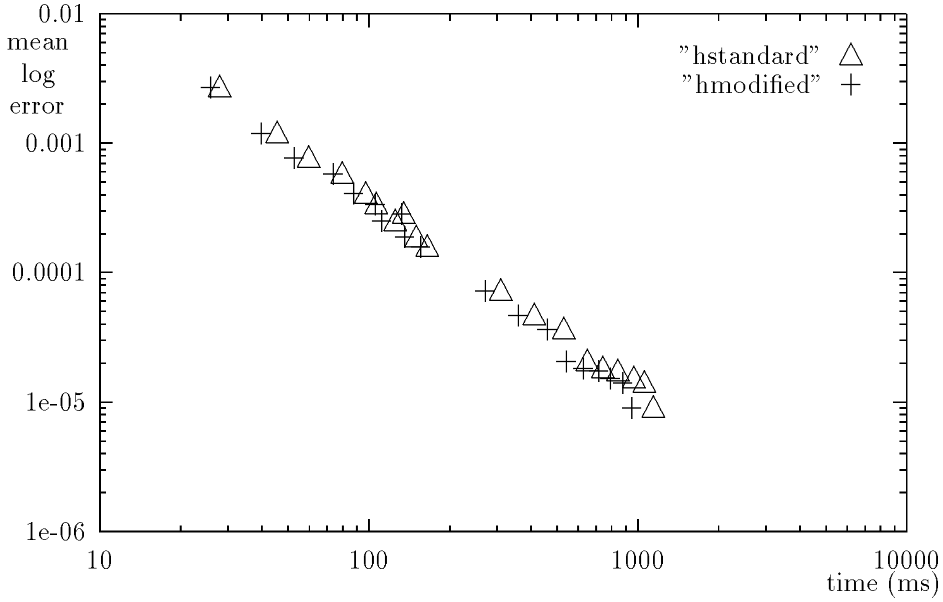


Figure 5.16: Results for the standard and modified stratified scheme for assessment functions uniformly drawn from $[0..0.1] \cup [0.9..1]$.

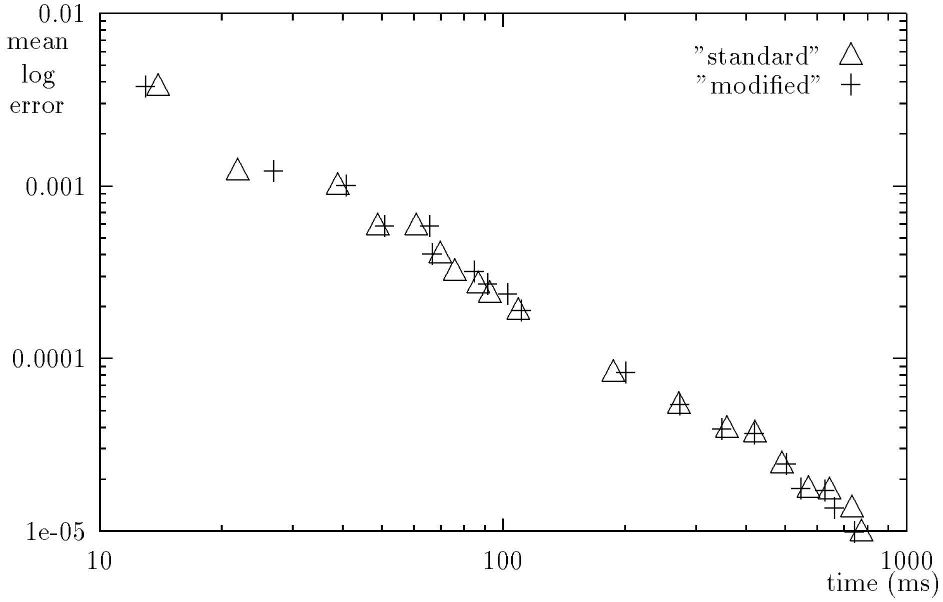


Figure 5.17: Results for the standard and modified stratified scheme.

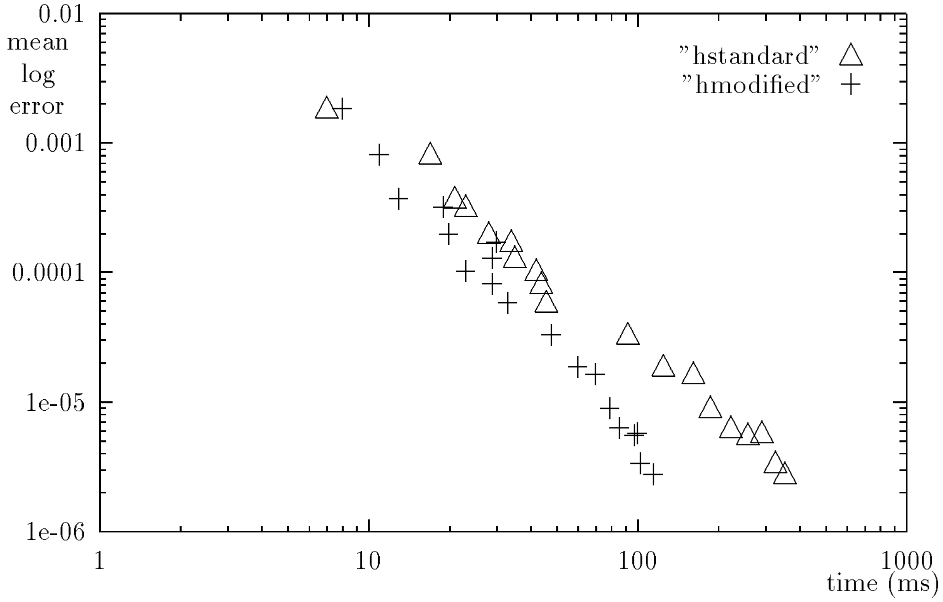


Figure 5.18: Results for the standard and modified stratified scheme for assessment functions uniformly drawn from $[0..0.1] \cup [0.9..1]$ for 25 nodes.

bilities, this effect is apparent; the saving in computation time is more than 30% for large sample sizes in Figure 5.18.

Conclusions and Further Research

In this chapter, we summarize our main contributions to the various parts of the lifecycle of a Bayesian belief network and point out some directions for further research. Following the division of the thesis, this chapter is divided into three sections. In Section 6.1, we consider the theory developed on conditional independence and dependence. In Section 6.2, we discuss learning Bayesian belief networks, which affects both network structure construction and assessment function estimation. In Section 6.3, we review probabilistic inference in Bayesian belief networks with stratified simulation.

6.1 Conditional Dependence

Conditional independence is one of the key concepts in knowledge-based systems. Without making assumptions about conditional independence, reasoning in a knowledge-based system would become intractable. Conditional dependence is the counterpart of conditional independence that has not been studied until now. In this thesis, we have presented a theory on conditional dependence that exactly mirrors the theory on conditional independence; dependency statements, dependency models, and dependency bases are for conditional dependence what independency statements, independency models and input lists are for conditional independence. We have given an axiomatic characterization of conditional dependence by defining dependency axioms. These axioms contain both dependency statements and independency statements.

We have defined graphical criteria, called coupling, for reading dependency statements from undirected graphs and directed acyclic graphs, where we assume that the graph is a minimal I-map of some independency model and all dependency statements read from the graph are in the complementary dependency model. Unfortunately, it is not possible to read all existing dependency statements from a graph. So, for some statements it cannot be deduced from the structure of the graph whether it is an independency or a dependency statement. These statements are interesting for inference tasks, such as planning and explanation since, if conditional independence holds, these tasks may be performed more efficiently. It may be interesting to design algorithms that use independencies like these.

Since algorithms that work on network structures and do not need consultation of the represented distribution tend to be more efficient in solving problems than algorithms that do need the represented distribution, it is desirable to have an efficient graphical representation of all independency statements and dependency statements. Thus far, graphical representations are either not capable of capturing all independencies or are not efficient in memory and thus not efficient for algorithms that perform on them [9, 42]. Therefore, it may be useful to put research effort in developing memory efficient graphical formalisms that do represent as many independencies as possible.

A graphical representation that has gained much popularity in the field of statistics is the *chain graph* [71, 119, 120]. A chain graph is a graph with both edges and arcs. A graphical criterion has been defined for reading independency statements from chain-graphs. An open question is whether a chain graph can be constructed from an independency model similar to the construction of undirected graphs and directed acyclic graphs. Further, it would be nice to have a characterization of the represented independency model as the closure of some input list, subjects that are addressed in [11]. Similar to undirected graphs and directed acyclic graphs, a dependency base and a coupling criterion for reading dependency statements from chain graphs might be defined. Since chain-graphs are a generalization of both undirected graphs and directed acyclic graphs, we expect that such a graphical criterion would generalize both coupling for undirected graphs and directed acyclic graphs.

When constructing a network structure of a Bayesian belief network, conditional independence information may be extracted from a domain expert. The conditional dependencies that can be read from the structure can be used to confront an expert with the consequences of his judgements about conditional independence. Care must be taken in selecting informative dependency statements. Methods for selecting dependency statements for evaluation of network structures are a subject for further research.

6.2 Learning

Learning Bayesian belief networks from data consists of three components: a quality measure to judge the quality of a network structure with respect to the database, a search algorithm to select reasonable candidate network structures, and a method of learning assessment functions. We address these subjects separately.

6.2.1 Quality Measures

We have discussed three major measures to judge the quality of a network structure and a database of cases: the Bayesian measure [24], information criteria, and a minimum description length (MDL) measure. The MDL measure can be considered to be a special case of an information criterion. We have shown that the MDL measure can be regarded as an approximation of the logarithm of the Bayesian measure in many cases. As a consequence, the MDL measure inherits all desirable properties of the Bayesian measure. However, we have shown that the Bayesian measure does not have the property of score equivalence, that is, it does not assign the same quality to all network structures that represent the same independency model. We have shown that both information criteria and the MDL measure do have this property, which may be exploited by search algorithms and in deriving theoretical results.

We have investigated both infinite-size database properties and finite-size database properties of the various quality measures under the assumption that there is no prior information. We have proved several optimality properties for infinite-size databases. Since these properties justify the use of the quality measures discussed for recovering causality as defined by Glymour and Spirtes [104], these results unify the results of learning of network structures based on quality measures and those of learning based on independency statements. In conclusion, all quality measures show the same desirable behavior for infinitely large databases.

However, the behavior of the measures differ for finite-size databases. We have derived an upper bound on the size of the largest parent set in a network structure with the highest quality for the various quality measures. We have shown that network structures learned with the Bayesian measure tend to contain far more arcs than those learned with the other two measures. This is due to information criteria and the MDL measure assigning a cost to each probability that has to be estimated to define the assessment functions in a Bayesian belief network with the network structure at hand. The Bayesian measure assigns costs only to probabilities over those configurations that appear in the database. Note that the behavior of the Bayesian measure can be altered by defining a proper prior distribution.

One of the main problems in learning Bayesian belief networks is that the number of parameters to be learned for defining a variable's assessment function, grows exponentially with the number of parents of the variable. The same problem arises when specifying these probabilities with the help of experts. To circumvent this problem, the noisy-or model was introduced [81], which requires only a linear number of parameters to define an assessment function. The noisy-or model has an intuitive interpretation and shows that in many cases the full model is too expressive at the cost of the necessity to specify a lot of numbers. Because restrictions in the expressiveness of the noisy-or model, generalization have been introduced [33, 56, 105] that all require fewer parameters than in the full model. To get as much information as possible out of a finite-size database, it may be useful to develop quality measures for learning Bayesian belief networks with noisy-or, generalized noisy-or, and other distributions as well. It appears that information criteria and the MDL measure generalize straightforwardly. For generalizing the Bayesian measure additional investigation is necessary.

In this thesis, we assumed that the variables in the domain are discrete. A short examination of various public available database learns that most databases contain one or more continuous variables. To make the direct application of the quality measures discussed possible, the ad-hoc technique commonly used is to quantize those continuous variables. Since there is a large freedom in selecting the number of quantization levels and values of those levels, intelligent automated support needs to be developed.

An alternative is to develop new quality measures for Bayesian belief networks with both continuous and discrete variables. This leads to a number of research questions. For Bayesian belief networks with discrete variables it is clear how the assessment functions are defined. However, continuous variables may follow, for example, a beta distribution, a gamma distribution, or a normal distribution. So, first of all, it should be determined which distribution the variables follow. Once the distributions are defined, information criteria and the MDL measure seem to generalize in a straightforward way as quality measures, but for the Bayesian measure some additional investigation is necessary. Efficient algorithms for estimation of the parameters of these distributions may need to be developed.

Often a database contains more information than just that the variables are discrete or continuous. Discrete variables for which there is an ordering on their values are very common. This extra knowledge may by informative but is neglected by the quality measures we discussed. New quality measures to be developed may take advantage of this kind of information.

One of the basic assumptions used in the derivation of all quality measures is that the ‘true’ joint probability distribution over the domain’s variables does not change over time. Thus, the ordering of the cases in the database is considered to be irrelevant, which may not be realistic. New measures need to be developed to deal with the very common case that the ‘true’ distribution does change in time.

Another frequently used basic assumption is that there are no missing values. However, in many databases omissions occur. Techniques currently available to handle missing values, like ‘filling in’ the gaps or treating a missing value as a separate variable-value, are computationally expensive, or lead to unsatisfactory results. Development of new measures might help in dealing with these computational problems.

6.2.2 Search Algorithms

To select a network structure with high quality, the space of all network structures is searched. We have shown that it is NP-hard to select a network structure with some optimality properties, provided an oracle is present that reveals conditional independence information. Therefore, in the general case it is unlikely that efficient algorithms exist for learning network structures from data.

This result justifies the use of search heuristics. We have discussed two greedy search heuristics known as K2 and B, and presented an efficient implementation of the latter. Both algorithms can be regarded as search heuristics that only look one step ahead. We have proposed generalizations of K2 and B in which more than one step is examined. Experiments suggest that such search algorithms indeed return network structure with higher quality when there is a high connectivity in the original network structure. However, the divergence between the learned and the original distribution does not decrease when more than one step is looked ahead.

Furthermore, we have shown how to apply the general-purpose combinatorial optimization algorithms tabu search, simulated annealing, and rejectionfree annealing to the problem of selecting network structures with high quality.

Our experimental results suggest that simple greedy search heuristics like K2 and B, with a two-step look-ahead suffice for returning good network structures. Though intelligent search algorithms may return a network structure with a slightly higher quality, the classifying potential of the Bayesian belief network with this network structure will not differ considerably from Bayesian belief networks with network structures returned by K2 or B. Therefore, we believe that it may not be worth the effort to consider other search algorithms for learning network structures.

Hidden variables are variables that are not in the database but have a large influence on the variables in the database. When quality measures are applied to the discovery of causal relations, it is very well possible that the variables in the database do not allow a causal explanation of the relation between variables because there are hidden variables. Some techniques for detecting the presence of hidden variables based on independency statements [104] have been developed. Similar techniques may be developed for search algorithms based on quality measures.

6.2.3 Learning Assessment Functions

Once a network structure with high quality has been selected, assessment functions need to be specified in order to obtain a complete Bayesian belief network. Assessment functions can be directly estimated from the database and the Bayesian belief network thus obtained may be used for probabilistic inference in a knowledge-based system.

An alternative approach is to select a set of network structures with high quality, estimate their respective assessment functions, and use this set of Bayesian belief networks by weighting the probabilities that they deliver by the quality of the network structures. When this set of network structures is appropriately chosen, the set of Bayesian belief networks can be represented efficiently by a single belief network, a process known as smoothing. Experiments have indicated that smoothing indeed decreases the divergence between the original and the learned distribution. We have shown that smoothing of assessment functions in belief networks can be performed effectively by incorporating smoothing in the search heuristics K2 and B.

This technique may also be applied when other distributions, such as the noisy-or model and its generalizations, beta, gamma, or normal distributions, are to be learned. Technical details to determine when and how to perform this efficiently are yet to be implemented. It is not clear whether a similar approach can successfully be applied to Bayesian belief networks with continuous variables, because a weighted sum of normal distributions need not be a distribution that can be efficiently represented.

6.3 Stratified Simulation

Inference in Bayesian belief networks is performed via calculation of conditional probabilities of events given observed events. Simulation is a general-purpose technique that can be applied to implement this task. We have presented a stratified simulation scheme for probabilistic inference in Bayesian belief networks. The scheme generates samples evenly distributed in the sample space and can be implemented efficiently. The scheme is computationally more efficient than the likelihood weighing scheme. Based on experiments, likelihood weighing is considered to be the most efficient scheme available so far [25, 97]. Due to the evenly-distributed samples, the scheme also returns a better approximation of probabilities. We have shown both theoretically and experimentally that the approximation of beliefs is not only performed faster but also that better approximations are returned than with existing schemes. We have investigated the effects of various optimizations specific to the scheme. Though for special network structures exact algorithms may outperform simulation schemes, our algorithm offers a robust general-purpose method for probabilistic inference without restrictions on the topology of networks.

We have given a theoretical analysis for the simple case that all variables are independent and identically distributed. A theoretical analysis for the general case where variables may take any dependence relation would be interesting for estimating the errors made in approximating beliefs. This analysis could be based on the use of the central limit theorem to approximate the logarithm of the probability of the different configurations as proposed by Druzdzel [34]. Such analysis could have practical significance because the mean and variance of this normal distribution can be efficiently calculated from the assessment functions in the Bayesian belief network [10] and that with observation of evidence new values of mean and variance can be incrementally updated at little computational cost. Closer investigation should make clear whether the normal approximation gives good results. Otherwise, extreme-value theory as addressed in [16] may be more appropriate.

The stratified simulation scheme is inherently based on discrete variables. However, many domains contain both discrete and continuous variables. To apply the stratified scheme to such domains, the following modifications may be made. By choosing an appropriate sampling distribution, the stratified simulation scheme can be applied to the discrete variables in the distribution and the forward sampling scheme to the continuous variables. The discrete variables get assigned a value first and the continuous variables get assigned a value later. Of course the score has to be adapted appropriately. When there are too many continuous variables, this scheme will not be more efficient than forward sampling since the sampling distribution is not optimally chosen, so more samples are necessary to give a representative sample. Experimental results will have to give insight in the question in how many continuous variables may appear in the network such that the stratified scheme is more appropriate than forward simulation.

The most probable explanation (MPE) is the configuration of variables that are not evidence variables with the highest probability given the observed values of the evidence variables. The MPE is used in control of knowledge-based systems and in abduction. We can apply the stratified simulation scheme to solve this problem by exploiting the observation that if we use the modified stratified scheme with m steps, we are sure to visit every configuration that has a probability larger than $1/m$ according to the sampling distribution. Let the sampling distribution be equal to the distribution of which we want to know the MPE. Then, if a configuration with probability larger than $1/m$ is found, the MPE is identified. Otherwise, we can run the algorithm again with a larger number of steps. This process can be repeated until the MPE has been found. If the sampling distribution is not equal to the distribution of which we want to know MPE, a correction factor need to be incorporated. Techniques for finding the k most probable explanations and finding the MPE of a subset of variables may be developed similarly.

Lauritzen and Spiegelhalter [72] introduced an exact inference algorithm based on passing messages in so-called junction trees. In this thesis, we applied the stratified simulation scheme for the approximation of marginal probabilities which can be regarded as a summation $\sum_{s \in S} f(s)/|S|$ where $|S|$ is large and f can be written as the product of a set of functions. In fact, we can apply this technique to any problem with the same structure. The messages that are sent during inference in junction trees can be written in a form shown above. Therefore, we expect that the stratified simulation scheme may be used to calculate an approximation of the messages. Especially when cliques are large, stratified simulation could result in a much faster inference algorithm at the cost of introducing a small error.

Bibliography

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.* John Wiley & Sons, Chichester, 1989.

[2] B. Abramson and A. Finizza. Using belief networks to forecast oil prices. *International Journal of Forecasting*, 7:299–315, 1991.

[3] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19:716–722, 1974.

[4] S. Andreassen, M. Wolbye, B. Falck, and S.K. Andersen. MUNIN - a causal probabilistic network for interpretation of electromyographic findings. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 366–372, 1987.

[5] I. Beinlich, H. Suermondt, R. Chavez, and G. Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings Artificial Intelligence in Medical Care*, pages 247–256, 1989.

[6] C. Berge. *Graphs and Hypergraphs.* North-Holland, Amsterdam, 1973.

[7] R.R. Bouckaert. Optimizing causal orderings for generating DAGs from data. In *Proceedings Uncertainty in Artificial Intelligence*, volume 8, pages 9–16, San Mateo (CA), 1992. Morgan Kaufmann.

[8] R.R. Bouckaert. Belief network construction using the minimum description length principle. In *Proceedings European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 41–48. Springer-Verlag, 1993.

[9] R.R. Bouckaert. IDAGs: A perfect map for any distribution. In *Proceedings European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 49–56. Springer-Verlag, 1993.

[10] R.R. Bouckaert, E. Castillo, and J.M. Gutiérrez. A modified simulation scheme for inference in Bayesian networks. *Submitted to the International Journal of Approximate Reasoning*, 1994.

[11] R.R. Bouckaert and M. Studený. Chain graphs: Semantics and expressiveness. *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 1995.

[12] H. Bozdogan. Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52:345–370, 1987.

[13] B.G. Buchanan and E.H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading (MA), 1984.

[14] W.L. Buntine. Theory refinement on Bayesian networks. In *Proceedings Uncertainty in Artificial Intelligence*, volume 7, pages 52–60, San Mateo (CA), 1991. Morgan Kaufmann.

[15] W.L. Buntine. A guide to the literature on learning graphical models. Obtainable by anonymous ftp from ack.arc.nasa.gov/pub/buntine/graphbib.ps.Z, 1994.

[16] E. Castillo, R.R. Bouckaert, J.M. Sarabia, and C. Solares. Error estimation in approximate bayesian belief network inference. In *Submitted to the conference Uncertainty in Artificial Intelligence*, volume 11, 1995.

[17] R.M. Chavez and G.F. Cooper. Hypermedia and randomized algorithms for medical expert systems. *Computer Methods and Programs in Biomedicine*, 32:5–16, 1990.

[18] D. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Proceedings 5th International Workshop on Artificial Intelligence and Statistics*. Springer-Verlag, 1995.

[19] H.L. Chin and G.F. Cooper. Bayesian belief network inference using simulation. In *Proceedings Uncertainty in Artificial Intelligence*, volume 5, pages 129–147. North-Holland, Amsterdam, 1989.

[20] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependency trees. *IEEE Transactions on Information Theory*, IT-14:462–467, 1968.

[21] S.L. Clove. Small-sample and large-sample statistical model selection criteria. In P. Cheeseman and R.W. Oldford, editors, *Lecture Notes in Statistics 89*. Springer-Verlag, 1994.

[22] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[23] G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from databases. In *Proceedings Uncertainty in Artificial Intelligence*, volume 7, pages 86–94, San Mateo (CA), 1991. Morgan Kaufmann.

[24] G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[25] S.B. Cousins, W. Chen, and N.E. Frisse. CABeN: A collection of algorithms for belief networks. Technical Report WUCS-91-25, Medical Informatics Laboratory, Washington University, St Louis, MO, 1991.

[26] S.B. Cousins, W. Chen, and N.E. Frisse. A tutorial introduction to stochastic simulation algorithms for belief networks. *Artificial Intelligence in Medicine*, 5:315–340, 1993.

[27] R. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14:1–13, 1946.

[28] P. Dagum and E. Horvitz. A Bayesian analysis of simulation algorithms for inference in belief networks. *Networks*, 23:499–516, 1993.

[29] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153, 1993.

[30] J.N. Darroch, S.L. Lauritzen, and T.P. Speed. Markov fields and log-linear interaction models for contingency tables. *Annals of Statistics*, 8:522–539, 1980.

[31] A.P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society B*, 41:1–31, 1979.

[32] A.P. Dempster. A generalization of Bayesian inference. *Journal of the Royal Statistical Society, Series B*, 30:325–339, 1968.

[33] F.J. Dièz. Parameter adjustment in Bayes networks. The generalized noisy OR-gate. In *Proceedings Uncertainty in Artificial Intelligence*, volume 9, pages 99–105, San Mateo (CA), 1993. Morgan Kaufmann.

[34] M. Druzdzel. Some properties of joint probability distributions. In *Proceedings Uncertainty in Artificial Intelligence*, volume 10, pages 187–194, 1994.

[35] J. Friedman. Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, 19:1–141, 1991.

[36] J.F. Fung and S.L. Crawford. Constructor: A system for the induction of probabilistic models. In *Proceedings American Association for Artificial Intelligence*, volume 8, pages 762–769, 1990.

[37] R. Fung and K. Chang. Weighting and integrating evidence for stochastic simulation in Bayesian networks. In *Proceedings Uncertainty in Artificial Intelligence*, volume 6, pages 209–219. North-Holland, Amsterdam, 1990.

[38] R. Fung and B. Del Favero. Backward simulation in Bayesian networks. In *Proceedings Uncertainty in Artificial Intelligence*, volume 10, pages 227–234, San Mateo (CA), 1994. Morgan Kaufmann.

[39] L. van der Gaag. Evidence absorption for belief networks. Technical Report RUU-CS-93-35, Department of Computer Science, Utrecht University, 1993.

[40] A. Gammerman and Z. Luo. Constructing causal trees from a medical database. Technical Report TR91002, Department of Computer Science, Heriot-Watt University, 1991.

[41] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman & Co., San Francisco, 1979.

[42] D. Geiger. Towards the formalization of informational dependencies. Technical Report CSD-880053, Cognitive Systems Laboratory, Computer Science Department, UCLA, 1988.

[43] D. Geiger and D. Heckerman. Learning Gaussian networks. In *Proceedings Uncertainty in Artificial Intelligence*, volume 10, pages 235–243, San Mateo (CA), 1994. Morgan Kaufmann.

[44] D. Geiger, A. Paz, and J. Pearl. Learning causal trees from dependence information. In *Proceedings American Association for Artificial Intelligence*, volume 8, pages 770–771, 1990.

[45] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Learning*, PAMI-6:721–741, 1984.

[46] F. Glover, C. McMillan, and B. Novick. Interactive decision software and computer graphics for architectural and space planning. *Annals of Operations Research*, 5:557–573, 1985.

[47] C. Glymour, P. Spirtes, and R. Scheines. Independence relations produced by parameter values in causal models. *Philosophical Topics*, 18:55–70, 1990.

[48] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.

[49] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Wesley, Reading (MA), 1989.

[50] J.W. Greene and K.J. Supowit. Simulated annealing without rejected moves. *IEEE Transactions on Computer-Aided Design*, CAD-5:221–228, 1986.

[51] E. Hannan and B. Quinn. The determination of the order of an autoregression. *Journal of the Royal Statistical Society B*, 41:191–195, 1979.

[52] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Technical report, Microsoft Research, 1994.

[53] D. Heckerman, D. Geiger, and D. Chickering. A stratified simulation scheme for inference in Bayesian belief networks. In *Learning Bayesian Networks: The Combination of Knowledge and Statistical Data*, volume 10, pages 293–301, San Mateo (CA), 1994. Morgan Kaufmann.

[54] D. Heckerman, E. Horvitz, and B. Nathwani. Towards normative expert systems: Part I, the pathfinder project. *Methods of Information in Medicine*, 31:90–105, 1992.

[55] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Proceedings Uncertainty in Artificial Intelligence*, volume 4, pages 149–163. North-Holland, Amsterdam, 1988.

[56] M. Henrion. Some practical issues in constructing belief networks. In *Proceedings Uncertainty in Artificial Intelligence*, volume 5, pages 161–173. North-Holland, Amsterdam, 1989.

[57] E. Herskovits. *Computer-Based Probabilistic-Network Construction*. PhD thesis, Section of Medical Informatics, University of Pittsburgh, 1991.

[58] E.H. Herskovits and G.F. Cooper. Kutató: An entropy-driven system for the construction of probabilistic expert systems from databases. In *Proceedings Uncertainty in Artificial Intelligence*, volume 6, pages 54–62. North-Holland, 1990.

[59] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39:345–351, 1987.

[60] L. Ingber. Very fast simulated re-annealing. *Journal of Mathematical and Computer Modeling*, 12:867–973, 1989.

[61] R.L. Kashyap. Optimal choice of ar and ma parts in autoregressive moving average models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4:99–104, 1982.

[62] H. Kiiveri, T.P. Speed, and J.B. Carlin. Recursive causal models. *Journal of the Australian Mathematical Society A*, 36:30–52, 1984.

[63] A.C. King and C.B. Read. *Pathways to Probability*. Holt, Rinehart and Winston, New York, 1963.

[64] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[65] U Kjearulff. A computational scheme forreasoning in dynamic probabilistic networks. In *Proceedings Uncertainty in Artificial Intelligence*, volume 8, pages 121–129, San Mateo (CA), 1992. Morgan Kaufmann.

[66] H.H. Ku and S Kullback. Approximating discrete probability distributions. *IEEE Transactions on Information Theory*, IT-15:444–447, 1969.

[67] S. Kullback. *Information Theory and Statistics*. John Wiley & Sons, New York, 1959.

[68] W. Lam. *Learning and Refining Bayesian Network Structures from Data*. PhD thesis, Department of Computer Science, University of Waterloo, 1994.

[69] W. Lam and F. Bacchus. Learning Bayesian belief networks, an approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

[70] P. Larrañaga and Y. Yurramendi. Structure learning approaches in causal probabilistic networks. In *Proceedings European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 227–232. Springer-Verlag, 1993.

[71] S.L. Lauritzen. Mixed graphical association models. *Scandinavian Journal of Statistics*, 16:273–306, 1989.

[72] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems (with discussion). *Journal of the Royal Statistical Society B*, 50:157–224, 1988.

[73] I. Matzkevich and B. Abramson. Deriving a minimal I-map of a belief network relative to a target ordering of its nodes. In *Proceedings Uncertainty in Artificial Intelligence*, volume 9, pages 159–165, San Mateo (CA), 1993. Morgan Kaufmann.

[74] A. Newell and H.A. Simon. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs (NJ), 1972.

[75] R. Nishii. Maximum likelihood principle and model selection when the true model is unspecified. *Journal of Multivariate Analysis*, 27:392–403, 1988.

[76] K.G. Olesen, S.L. Lauritzen, and F.V. Jensen. aHUGIN: A system creating adaptive causal probabilistic networks. In *Proceedings Uncertainty in Artificial Intelligence*, volume 8, pages 223–229, San Mateo (CA), 1992. Morgan Kaufmann.

[77] R. Otten and L. van Ginneken. *The Annealing Algorithm*. Kluwer Academic Publishers, Boston, 1989.

[78] Dagum. P., A. Galper, and E. Horvitz. Dynamic network models for forecasting. In *Proceedings Uncertainty in Artificial Intelligence*, volume 8, pages 41–48, San Mateo (CA), 1992. Morgan Kaufmann.

[79] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

[80] J. Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32:241–288, 1987.

[81] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo (CA), 1988.

[82] J. Pearl, D. Geiger, and T. Verma. The logic of influence diagrams. In R.M. Oliver and J.Q. Smith, editors, *Influence Diagrams, Belief Nets and Decision Analysis*, pages 67–87. John Wiley & Sons, 1990.

[83] J. Pearl and A. Paz. Graphoids: A graph based logic for reasoning about relevance relations. In B. Du Boulay, D. Hogg, and L. Steels, editors, *Advances in Artificial Intelligence II*, pages 357–363. North-Holland, Amsterdam, 1987.

[84] J. Pearl and T. Verma. A theory of inferred causation. In A. Fikes and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation*, pages 441–452, 1991.

[85] D. Raj. *Sampling Theory.* McGraw-Hill, New York, 1968.

[86] G. Rebane and J. Pearl. The recovery of causal polytrees from statistical data. In *Proceedings Uncertainty in Artificial Intelligence*, volume 3, pages 222–228. North-Holland, Amsterdam, 1987.

[87] R. Reiter. Nonmonotonic reasoning. *Annual Review of Computer Science*, 2:147–186, 1987.

[88] J. Rissanen. A universal data compression system. *IEEE Transactions on Information Theory*, IT-29:656–664, 1983.

[89] J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

[90] J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society B*, 49:223–239, 1987.

[91] R.D. Robinson. Counting unlabeled acyclic digraphs. In *Proceedings Australian Conference on Combinatorial Mathematics*, volume 5, pages 28–43, 1976.

[92] R.Y. Rubinstein. *Simulation and Monte Carlo Methods.* Wiley, New York, 1981.

[93] L.J. Savage. *The Foundations of Statistics.* Dover, New York, 1954.

[94] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[95] R.D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[96] R.D. Shachter. An ordered examination of influence diagrams. *Networks*, 20:535–563, 1990.

[97] R.D. Shachter and M. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Proceedings Uncertainty in Artificial Intelligence*, volume 6, pages 221–231. North-Holland, Amsterdam, 1990.

[98] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, 1976.

[99] G. Shafer and J. Pearl, editors. *Readings in Uncertain Reasoning.* Morgan Kaufmann, San Mateo (CA), 1990.

[100] E.H. Shortliffe. *MYCIN: A Rule-Based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection.* PhD thesis, Stanford Artificial Intelligence Laboratory, Stanford University, 1974.

[101] D.J. Spiegelhalter, A.P. Dawid, S.L. Lauritzen, and R.G. Cowell. Bayesian analysis in expert systems. *Statistical Science*, 8:219–283, 1993.

[102] D.J. Spiegelhalter and S.L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.

[103] P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal structures. *Social Science Computer Review*, 9:62–72, 1991.

[104] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* Springer-Verlag, New York, 1993.

[105] S. Srinivas. A generalization of the noisy-OR model. In *Proceedings Uncertainty in Artificial Intelligence*, volume 9, pages 208–215, San Mateo (CA), 1993. Morgan Kaufmann.

[106] M. Studený. Multiinformation and the problem of characterization of conditional-independence relations. *Problems of Control and Information Theory*, 18:3–16, 1989.

[107] M. Studený. Formal properties of conditional independence in different calculi of AI. In *Proceedings European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 341–348. Springer-Verlag, 1993.

[108] H.J. Suermondt and G.F. Cooper. Probabilistic inference in multiply connected belief networks using loop cutsets. *Journal of Approximate Reasoning*, 4:283–306, 1990.

[109] J. Suzuki. A construction of Bayesian networks from databases based on the MDL principle. In *Proceedings Uncertainty in Artificial Intelligence*, volume 9, pages 266–273, San Mateo (CA), 1993. Morgan Kaufmann.

[110] H.H. Szu and R.L. Hartley. Fast simulated annealing. *Physical Letters A*, 122:157–162, 1987.

[111] E. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[112] A. Tversky and D. Kahneman. Extensional versus intuitive reasoning: The conjunction fallacy in probability judgement. *Psychological Review*, 4:293–315, 1983.

[113] V. Černy. A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.

[114] T. Verma and J. Pearl. Causal networks: Semantics and expressiveness. In *Proceedings Uncertainty in Artificial Intelligence*, volume 4, pages 352–359. North-Holland, Amsterdam, 1988.

[115] T. Verma and J. Pearl. An algorithm for deciding if a set of observed independencies has a causal explanation. In *Proceedings Uncertainty in Artificial Intelligence*, volume 8, pages 323–330, San Mateo (CA), 1992. Morgan Kaufmann.

[116] W.A.T. Wan Abdullah. Seeking global minima. *Journal of Computational Physics*, 110:320–326, 1994.

[117] D. Wedelin. *Efficient Algorithms for Probabilistic Inference, Combinatorial Optimization and the Discovery of Causal Structure from Data.* PhD thesis, Department of Computer Sciences, Chalmers University of Technology Göteborg, Sweden, 1993.

[118] N. Wermuth and S.L. Lauritzen. Graphical and recursive models for contingency tables. *Biometrika*, 72:537–552, 1983.

[119] N. Wermuth and S.L. Lauritzen. On substantive research hypothesis, conditional independence graphs and graphical chain models. *Journal of the Royal Statistical Society B*, 52:21–50, 1990.

[120] J. Whittaker. *Graphical Models in Applied Mathematical Multivariate Statistics.* John Wiley & Sons, Chichester, 1990.

[121] S.S. Wilks. *Mathematical Statistics.* John Wiley & Sons, New York - London, 1962.

[122] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1:3–28, 1978.

Curriculum vitae

Remco Ronaldus Bouckaert

27 Januari 1967

Geboren te Haarlem

September 1979 - Mei 1985

VWO Nieuw Lyceum Hilversum

September 1985 - Februari 1991

Technische Universiteit Eindhoven
Propaedeuse elektrotechniek 1986
Doctoraal informatietechniek 1991

Maart 1991 - Maart 1995

AIO aan de Universiteit Utrecht, vakgroep informatika

Samenvatting

In het dagelijks leven is het redeneren met onzekerheden gebruikelijker dan het redeneren zonder. Bayesiaanse belief netwerken bieden een wiskundig correct formalisme om onzekerheid te representeren en op efficiënte wijze mee te redeneren. Een Bayesiaanse belief netwerk bestaat uit twee delen.

Ten eerste bestaat een belief netwerk uit een een gerichte graaf zonder lussen: de *netwerkstructuur*. Voor elke variabele waarmee we willen redeneren is er een knoop in de graaf. We zullen de termen knoop en variabele dan ook door elkaar gebruiken. Figuur 0.1 laat een eenvoudig belief netwerk zien voor een klein medisch domein met daarin de leeftijd van een patient (a), de behoefte aan een bril (g), of het zicht beter wordt als de patient knippert (v) en of de patient klachten heeft over zijn zicht (s). Als er een directe afhankelijkheid tussen twee knopen is, dan zijn deze knopen verbonden met een pijl. Intuitief geeft de richting van de pijl een causale invloed aan. Bijvoorbeeld in Figuur 0.1 geeft de pijl van a naar g weer dat de leeftijd een indicatie is dat de patient een bril nodig heeft.

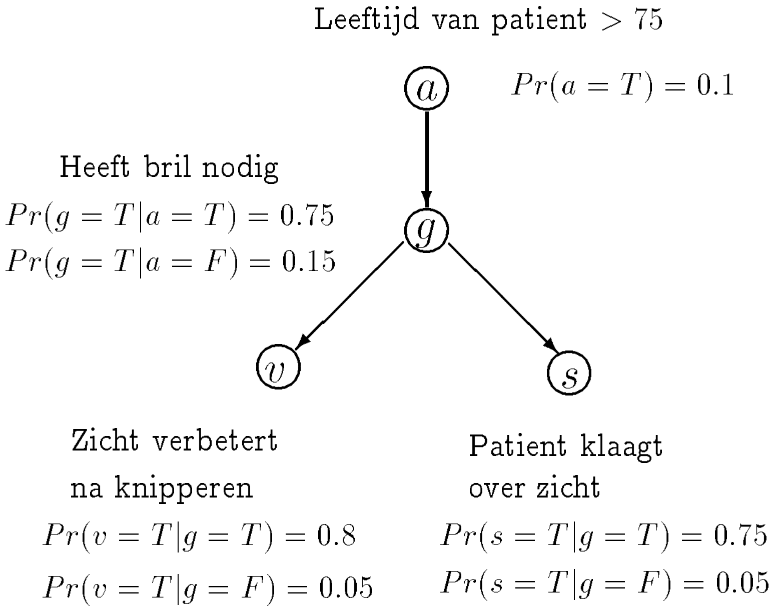


Figure 0.1: Eenvoudig belief netwerk.

Ten tweede bestaat een Bayesiaanse belief netwerk uit een verzameling *conditionele kansverdelingen*, één voor elke knoop geconditioneerd op de ouder knopen. In Figuur 0.1 zijn deze kansverdelingen naast iedere knoop weergegeven. Al deze kansverdelingen samen definiëren een kansverdeling over de hele verzameling van variabelen.

In dit proefschrift behandelen we drie essentiële onderdelen in de levens cyclus van een belief netwerk; de door de netwerkstructuur gerepresenteerde afhankelijkheden, het automatisch leren van belief netwerken uit databases, en het redeneren met behulp van belief netwerken.

Gerepresenteerde Afhankelijkheden

De efficiëntie van veel algoritmen die worden toegepast op belief netwerken is gebaseerd op het feit dat vaak grote verzamelingen variabelen irrelevant zijn voor het redeneren met andere variabelen. Irrelevantie kan worden geformaliseerd met het begrip *conditionele onafhankelijkheid*; de verzamelingen variabelen X en Y zijn conditioneel onafhankelijk gegeven de verzameling variabelen Z, als voor alle waarden van de variabelen in X, Y en Z geldt, dat het kennen van de waarden van variabelen in Y irrelevant is om iets over de waarden van de variabelen X af te leiden als we de waarden van de variabelen Z al kennen. Bijvoorbeeld, als bekend is dat een patient een bril nodig heeft, dan is het niet relevant of de patient klachten heeft over zijn zicht om iets over de leeftijd van de patient af te leiden.

Een *onafhankelijkheidsmodel* is een verzameling conditionele onafhankelijkheden. Een veel voorkomend onafhankelijkheidsmodel is het *graphoide onafhankelijkheidsmodel*: het is gesloten is onder afleiding met behulp van de zogenaamde *graphoide axioma's*. Voor deze modellen is veel theorie ontwikkeld:

- Er zijn criteria om conditionele onafhankelijkheden af te lezen uit netwerkstructuren.
- Er bestaat een relatie tussen gerepresenteerde conditionele onafhankelijkheden in een netwerkstructuur en die gerepresenteerd in de kansverdeling behorend bij een belief netwerk.
- Er zijn efficiënte methode om netwerkstructuren te construeren uit een onafhankelijkheidsmodel M^I die een maximale deelverzameling van M^I representeert, zogenaamde *minimale I-maps*.

Helaas kan niet elk onafhankelijkheidsmodel precies gerepresenteerd worden met behulp van een graaf; er zullen er vaak conditionele onafhankelijkheden zijn die wel in het onafhankelijkheidsmodel zitten maar niet door de graaf gerepresenteerd worden.

In dit proefschrift wordt een theorie ontwikkeld over *conditionele afhankelijkheid*, het tegengestelde van conditionele onafhankelijkheid; X en Y zijn conditioneel afhankelijk gegeven Z als X en Y niet conditioneel onafhankelijk zijn. Een *afhankelijkheidsmodel* is een verzameling conditionele afhankelijkheden. Met elk onafhankelijkheidsmodel kan een afhankelijkheidsmodel geassocieerd worden zodanig dat elk triple (X, Z, Y) in één van beide modellen zit. Stel dat M^I een graphoide onafhankelijkheidsmodel is en M^D het complementaire afhankelijkheidsmodel. Een aantal axioma's is opgesteld waaronder een groot aantal afhankelijkheidsmodellen gesloten is. In dit proefschrift is een criterium ontwikkeld om een groot aantal conditionele afhankelijkheden uit M^D af te lezen uit een netwerkstructuur die een minimale I-map van M^I is. Helaas kan niet van elk triple geïdentificeerd worden of het een conditionele onafhankelijkheid of een conditionele afhankelijkheid is.

Automatisch Leren van Belief Netwerken

Een belief netwerk bestaat uit twee componenten; een netwerkstructuur en een verzameling conditionele kansverdelingen. Het leren van een belief netwerk van een database kan eveneens worden opgedeeld in tweeën; het leren van een netwerkstructuur en het leren van de conditionele kansverdelingen.

Een netwerkstructuur kan worden geleerd met behulp van een kwaliteitsmaat, een maat die aangeeft hoe goed een netwerkstructuur bij een database past. In dit proefschrift zijn de eigenschappen van verschillende kwaliteitsmaten onderzocht, te weten de Bayesiaanse maat, de MDL maat en informatiecriteria. Er wordt aangetoond dat de MDL maat als een goede benadering van de Bayesiaanse maat gezien kan worden voor veel databases. We hebben laten zien dat door een informatiecriterium aan elke netwerkstructuur die dezelfde conditionele onafhankelijkheden representeert, dezelfde kwaliteit wordt toegekend. Deze eigenschap geldt ook voor de MDL maat. De Bayesiaanse maat heeft deze eigenschap niet. We hebben voor verscheidene situaties laten zien dat alle drie de kwaliteitsmaten de gewenste eigenschappen hebben voor oneindig grote databases. Voor eindig grote databases hebben we bovengrenzen afgeleid voor het aantal ouders van een knoop in de beste netwerkstructuur. Het blijkt dat de Bayesiaanse maat vaak netwerken met knopen met veel meer ouders prefereert dan de andere maten.

Aangezien het aantal mogelijke netwerkstructuren exponentieel groeit met het aantal knopen, is het niet mogelijk een kwaliteitsmaat toe te passen op alle mogelijke netwerkstructuren. Er is een zoekalgoritme nodig dat netwerkstructuren selecteert waarvan verwacht kan worden dat ze een hoge kwaliteit hebben. In dit proefschrift laten we zien dat het selecteren van netwerkstructuren met zekere optimaliteitseigenschappen NP-moeilijk is. Dit betekent dat het zeer waarschijnlijk is dat er geen efficiënt algoritme bestaat om zulke netwerkstructuren te selecteren. Daarom maken we gebruik van heuristieken. We laten zien hoe de bekende heuristieken K2 en B gegeneraliseerd kunnen worden en hoe 'simulated annealing', 'tabu search' en 'rejectionfree annealing' toegepast kunnen worden op dit probleem.

Als eenmaal een netwerkstructuur met een hoge kwaliteit geselecteerd is, dan kunnen de conditionele kansverdelingen geleerd worden door rechtstreeks de kansen uit de database te schatten. Een alternatieve methode is om een verzameling netwerkstructuren met hoge kwaliteit te selecteren, en voor deze verzameling structuren kansen te schatten. De verzameling aldus verkregen netwerken representeren een kansverdeling die een met de kwaliteit gewogen gemiddelde is van de netwerken in de verzameling. Voor speciale verzamelingen kan deze kansverdeling door één enkel belief netwerk gerepresenteerd worden. In dit proefschrift laten we zien dat het construeren van zo een belief netwerk efficiënt gedaan kan worden tijdens het uitvoeren van de heuristieken K2 en B.

Redeneren met Belief Netwerken

Redeneren met belief netwerken is gebaseerd op het berekenen van conditionele kansen. Met het belief netwerk uit Figuur 0.1 kan bijvoorbeeld de kans berekend worden dat een patient een bril nodig heeft wanneer bekend is dat de patient ouder is dan 75 jaar. Een algemeen toepasbare techniek die voor deze berekening gebruikt kan worden is simulatie. In dit proefschrift wordt een simulatiealgoritme gepresenteerd voor het redeneren met belief netwerken gebaseerd op een populaire statistische techniek, stratificatie. Het stratificatiealgoritme genereert steekproeven die gelijkmatig over de sample ruimte verdeeld zijn. Het algoritme kan efficiënt geïmplementeerd worden. We laten zowel theoretisch als experimenteel zien dat het algoritme zowel een kortere rekentijd als een betere kansschatting oplevert dan het ‘likelihood weighing’ algoritme, het meest efficiënte algoritme dat tot dusver bekend was. Verscheidene optimalisaties van ons algoritme hebben we onderzocht. Hoewel andere algoritmen voor speciale netwerkstructuren efficiënter kunnen zijn, biedt stratificatie een algemeen toepasbaar robuust algoritme waarvan de efficiëntie niet door de netwerkstructuur beïnvloed wordt.

Index