Stellingen

behorend bij het proefschrift

Problem-directed Decomposition
of
Bayesian Belief Networks

Erica van de Stadt
12 september 1995
I

De decompositie van een Bayesian belief network, verkregen door toepassing van de in dit proefschrift voorgestelde methode, doet recht aan de op zichzelf weerstrevende eisen van algemeenheid en efficientie.
(dit proefschrift, Hoofdstuk 4)

II

De voorgestelde decompositiemethode kan het meest succesvol worden toegepast als de structuur van het betreffende Bayesian belief network gekarakteriseerd is door clusters die onderling verbonden zijn in een polytree.
(dit proefschrift, Hoofdstuk 5)

III

Het ontbreken van consensus over een indeling van bestaande Bayesian belief networks in klassen op grond van topologische eigenschappen bemoeilijkt de evaluatie van methoden die tot doel hebben het aantal rekenoperaties uitgevoerd tijdens inferentie te verminderen.
(dit proefschrift, Hoofdstuk 5)

IV

Met de introductie van beslisnetwerken\(^1\) is een methode verkregen waarin het plannen van acties verweven is met het verwerken van informatie zodra ze beschikbaar komt. Dit garandeert dat de geplande acties voortdurend toegesnptst zijn op de actuele situatie.
(dit proefschrift, Hoofdstuk 6)


V

Zodra het ondernemen van acties naast positieve ook negatieve gevolgen kan hebben behoren diagnose en planning verweven processen te zijn.
VI

Het toepassing van kennisystemen in complexe domeinen zal pas daadwerkelijk ingang vinden als ze geheel of gedeeltelijk automatisch geconstrueerd kunnen worden mits dit niet leidt tot “black box” systemen.

VII

Worden intelligent gedrag en cognitieve ontwikkeling losgekoppeld, dan kan worden gesteld dat de meest eenvoudige rekenmachine intelligent gedrag vertoont.

VIII

Het getuigt van realiteitszin om bij het combineren van (betaald) werk en de zorg voor een gezin te kiezen voor deeltijdfuncties.

IX

Gebruikmaken van geslachtskeuzetechnieken om emotionele redenen vergroot de kans op desillusies.

X

Discriminatie kan niet positief zijn.

XI

Bij de selectie van jonge paarden bestemd voor de springsport is enige mate van bodemschuwheid een pré.
Problem-directed Decomposition
of
Bayesian Belief Networks

Erica van de Stadt
Problem Directed Decomposition
of
Bayesian Belief Networks

Proefschrift ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.ir.K.F.Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op dinsdag 12 september 1995 te 10.30 uur

door

Elisabeth Carolien VAN DE STADT,
doctorandus in de informatica,
geboren te Kaduna (Nigeria)
Problem-directed decomposition of Bayesian belief networks / Elisabeth Carolien van de Stadt. - [S.l. : s.n.]. - I11
Thesis Technische Universiteit Delft. - - With ref. -
With summary in Dutch.
ISBN 90-9008529-7
Subject headings: Bayesian belief networks ; efficient inference /
Bayesian belief networks : planning.
Aan Jelle en Anouk.
Aan Han, Hetty en Marjet.
## Contents

<table>
<thead>
<tr>
<th>Summary</th>
<th>ix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Early approaches</td>
<td>2</td>
</tr>
<tr>
<td>1.1.1 Idiot Bayesian approach</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 Rule-based approach</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Extensional and intensional</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Bayesian belief networks</td>
<td>6</td>
</tr>
<tr>
<td>1.3.1 Singly connected networks</td>
<td>6</td>
</tr>
<tr>
<td>1.3.2 Multiply connected networks</td>
<td>8</td>
</tr>
<tr>
<td>1.3.3 Computational complexity</td>
<td>8</td>
</tr>
<tr>
<td>1.4 Concluding remarks</td>
<td>9</td>
</tr>
<tr>
<td>1.5 Research objectives</td>
<td>9</td>
</tr>
<tr>
<td>1.6 Scope of the thesis</td>
<td>10</td>
</tr>
<tr>
<td>2 Graphical models</td>
<td>13</td>
</tr>
<tr>
<td>2.1 Dependency models and independency statements</td>
<td>14</td>
</tr>
<tr>
<td>2.2 Dependency models characterized by undirected graphs</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Dependency models characterized by directed acyclic graphs</td>
<td>16</td>
</tr>
<tr>
<td>2.3.1 d-separation</td>
<td>17</td>
</tr>
<tr>
<td>2.3.2 Directed acyclic graph representations of probabilistic dependency models</td>
<td>20</td>
</tr>
<tr>
<td>2.4 Discussion</td>
<td>21</td>
</tr>
<tr>
<td>3 Bayesian belief networks</td>
<td>23</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>23</td>
</tr>
</tbody>
</table>
5.4.4 Performed experiments ....................................... 85
5.5 Results and discussion ........................................... 87
  5.5.1 The data from the experiments ................. 87
  5.5.2 Tendencies ........................................... 88
5.6 Conclusion and remarks ......................................... 94
5.7 Two real networks .............................................. 95
  5.7.1 CNMF ............................................... 96
  5.7.2 Results and discussion .............................. 97
  5.7.3 Alarm .............................................. 102
  5.7.4 Closing comparison .................................. 111

6 Decision networks .............................................. 113
  6.1 Introduction .............................................. 113
  6.2 Approached to the planning of actions .............. 114
  6.3 Decision networks ......................................... 118
    6.3.1 From Bayesian belief network to decision network .... 118
    6.3.2 Evaluating decision networks ................... 118
    6.3.3 Computational complexity ....................... 122
  6.4 Decision networks and decision trees ............... 125
    6.4.1 The single-test/single-treat decision network ...... 127
    6.4.2 The single-test/single-treat decision tree .......... 127
    6.4.3 Analysis ......................................... 129
    6.4.4 Comparison ....................................... 130
  6.5 Discussion and conclusion .................................. 132
  6.6 The CNMF decision network ............................... 133
    6.6.1 Extension to the decision network ............. 133
    6.6.2 Problem instance mapping and problem-directed decompo-
        sition ............................................. 136
    6.6.3 Concluding remarks ................................ 140

7 Discussion and directions for further research .......... 145
  7.1 Discussion .............................................. 145
  7.2 Direction of further research ........................... 150

Bibliography .................................................. 153

A Triangulation ............................................... 159
B  Accumulated data from the experiments  165

C  Additional specifications for the real-life networks CNMF and Alarm  171

D  Bending in the test strategy  175

  Samenvatting  181

  Acknowledgements  187

  Curriculum Vitae  189
Notations and abbreviations

$U$ finite set of propositional variables

$\mathcal{M}$ dependency model

$X, Y, Z$ disjunct subsets of $U$

$I(X, Y, Z)_\mathcal{M}$ independency statement

$G = (V(G), A(G))$ $G$ is a directed acyclic graph

$V(G)$ vertex set of $G$

$V_i$ element of $V(G)$, and represents a propositional variable

$Sp(V_i)$ state space of $V_i$

$Sp(V_i) = \{v_i, \neg v_i\}$ $V_i$ is a binary variable which is either true or false

$A(G)$ arc set of $G$

$(V_i, V_j) \in A(G)$ arc pointing from $V_i$ to $V_j$

$\rho_G(V_i)$ set of parents of $V_i$ in $G$

$\sigma_G(V_i)$ set of children of $V_i$ in $G$

$G_m$ moral graph associated with $G$

$G_u$ triangulated graph associated with $G$

$Cl(G_u)$ clique set of $G_u$

$Clq_i$ a clique

$\varphi_{Clq_i}$ potential function of $Clq_i$
\[ C = (\text{Cl}(G), E(G)) \quad \text{maximum weight spanning tree} \]
\[ T(B) = (C, \emptyset) \quad \text{junction tree associated with } B \]
\[ G = (V(G), E(G)) \quad G \text{ is an undirected graph} \]
\[ \text{Adj}_G(V_i) \quad \text{set of neighbors in } G \]
\[ (V_i, V_j) \in E(G) \quad \text{edge between } V_i \text{ and } V_j \]
\[ (X \mid Z \mid Y)_G \quad \text{graph separation assertion} \]
\[ B = (G, \mathcal{P}) \quad \text{Bayesian belief network} \]
\[ \mathcal{P} \quad \text{set of all (conditional) probability assessment functions} \]
\[ \mathcal{P}(V_i) \quad \text{assessment function assigned to root vertex } V_i \]
\[ \mathcal{P}(V_i \mid C_{\rho_0(V_i)}) \quad \text{conditional assessment function assigned to vertex } V_i \]
\[ \text{Bel}(V_i) \quad \text{updated probability of vertex } V_i, \text{ given the evidence introduced in } B \]
\[ (\mathbf{E}, \mathbf{Q}) \quad \text{problem instance} \]
\[ \mathbf{E} \quad \text{set of pieces of evidence} \]
\[ \mathbf{Q} \quad \text{set of query variables} \]
\[ G^E \quad \text{absorbed graph} \]
\[ V^E \quad \text{virtual evidence vertices} \]
\[ B^E = (G^E, \mathcal{P}^E) \quad \text{absorbed Bayesian belief network} \]
\[ G^{(\mathbf{E}, \mathbf{Q})} \quad \text{relevant graph of } G \text{ with respect to problem instance } (\mathbf{E}, \mathbf{Q}) \]
\[ JV \quad \text{set of joining vertices} \]
\[ J^{(\mathbf{E}, \mathbf{Q})} \quad \text{rest graph} \]
\[ \mathbf{I}^* \quad \text{strongly independent components} \]
\[ G_V \quad \text{graph induced on } G \text{ by the vertex set } V \subseteq V(G) \]
$\mathcal{P}^G$  set of all (conditional) probability assessment function induced on $G$

$B_G$  Bayesian belief network induced on the induced graph $G$

$B^{(E,Q)}$  relevant (Bayesian belief) network

$\tilde{B}^{(E,Q)}$  rest (Bayesian belief) network

$T^u(B^{(E,Q)})$  upgraded junction tree

$D = (B, (D, U), U)$  decision network

$D$  set of decision vertices

$D_i$  treatment decision vertex

$T_i$  test decision vertex

$U$  set of utility vertices

$U$  set of utility vectors

$U(U_i)$  utility vector assigned to utility vertex $U_i$

$V^{(\cdot)}(U_i)$  value function of $U_i$

$c_i^*$  optimal test decision(s)

$c_{tr}^*$  optimal treatment decision(s)

$F_{tr}(c_{tr}^*)$  treatment utility function

$F_{tr}^{R_{ti}=pos}(c_{tr}^*)$  treatment utility function after a positive test result has been instantiated

$F_{t}(c_t^*)$  test utility function
Summary

At present, within the AI community the necessity for methods to represent and manipulate uncertain and incomplete knowledge has largely been acknowledged. In this thesis, the Bayesian belief network formalism is considered as one possible approach to meet the above requirements.

The Bayesian belief network consists of a quantified directed acyclic graph. The vertices of this graph correspond to (random) variables, and they for their part represent concepts from the knowledge domain. The arcs represent relations among the variables enabling mutual influences. The absence of arcs represents independency relations valid for the variables discerned in the domain.

Uncertainty is represented by conditional probability distributions that quantify the strength of influence, and a priori probability distributions that quantify initial probabilities of the hypotheses. Further, the propagation of probabilities is consistent with probability theory. The computational complexity of the propagation of probabilities in Bayesian belief networks is inherent exponential. More specifically, probabilistic inference in Bayesian belief networks is a NP-hard problem. This classification hinders the application of Bayesian belief networks in complex domains.

In this thesis, a new decomposition method for Bayesian belief networks is proposed. The decomposition method aims at bringing under control the computational complexity of the probabilistic inference. The method departs from a given Bayesian belief network and a given problem instance. A problem instance consists of two sets of variables: the evidence variables, values of which are known, and the query variables. We are interested in the set of individual probability distributions of the query variables updated in the light of the evidence. This set is called the solution of a problem instance.

The proposed decomposition method decomposes a given Bayesian belief network into the relevant net and the rest net. The relevant net is of direct impor-
tance to the determination of the solution of the problem instance and for the rest net this is not the case. Loosely speaking, the relevant net consists of all paths that in the original Bayesian belief network connect an evidence variable with a query variable. The number of calculations necessary for the determination of the solution of the problem instance is restricted because the impact the evidence variables have on the query variables only need to be propagated in the relevant net. In order to diminish the number of computations, probabilistic inference is restricted to the relevant net.

Several decomposition methods are known from the literature. However, these methods do not possess general applicability and are not able to handle changes in the problem instance. The development of the new decomposition method is guided by the requirements of general applicability, flexibility and exact probabilistic inference.

As a result of the general applicability requirement, the method for the propagation of probabilities in junction trees has been chosen. The junction tree inference method can, in principle, be applied in order to propagate probabilities in any multiply connected Bayesian belief network.

The flexibility requirement amounts to the simple adaptation of the decomposition after the problem instance has changed. Actually, this means without the need to reconsider the original Bayesian belief network and reconstruction of the associated junction tree. For example, the junction tree of the relevant net might need to be temporarily extended by the addition of parts of the junction tree of the rest net.

The requirement of exact probabilistic inference dictates that the solution of a given problem instance as calculated in the relevant net must be identical to the solution of the problem instance as calculated in the original Bayesian belief network.

The effect of applying the proposed decomposition method has been evaluated. The best-case and worst-case complexity of the proposed decomposition method are easily identified by a theoretical analysis. For the average behavior this is not possible.

Based on a global idea of the topological properties of Bayesian belief networks, a number of hypotheses about the average behavior behavior have been formulated. Subsequently, an experimental evaluation was conducted in order to verify these hypotheses. Because there is no consensus about a classification of
the few existing Bayesian belief networks based on their topological properties, we chose to perform the experiments with classes of randomly generated Bayesian belief networks. The results of these experiments confirm the hypotheses. The influence of the connectivity density of the Bayesian networks, however, is remarkable. This observation is explained by the nature of the randomly generated Bayesian belief networks.

Real-life Bayesian belief networks can be included in each of the considered classes. Therefore, general conclusions about the behavior of the proposed decomposition method in practice cannot be drawn. This problem is existent in the experimental evaluation of any method in which behavior relies heavily on the topological properties of the Bayesian belief network. Nevertheless, for the time being, using randomly generated networks will be the only possibility to enable an experimental evaluation of the average behavior.

The positive effect of applying the proposed decomposition method can also be demonstrated in existing Bayesian belief networks. Existing Bayesian belief networks are not without further ado suitable for an experimental evaluation, because often during construction, efforts are made to simplify the topology in order to restrict the computational complexity.

Experiments are conducted with two existing Bayesian belief networks; the Central Neural Muscular Failure (CNMF) network and the Alarm network. Both networks consist of approximately the same number of vertices, and simplifications for the benefit of inference speed are not made.

On the scale from best-case to worst-case complexity, the application of the decomposition method in the CNMF network is located near the best case, while the application of the decomposition method in the Alarm model is located near the worst case. This significant difference is largely determined by the topological properties of the respective networks. The complexity of the CNMF network is low compared to the complexity of the Alarm network because fewer vertices are contained in one or more loops. The CNMF network possesses almost a polytree topology; the two non-overlapping loops contain approximately a quarter of the total number of vertices and can be seen as local condensation of arcs in the topology (clusters). In the decomposition, the two clusters are separated such that the relevant net is both less complex and less extensive than the original Bayesian belief network. The Alarm network contains six loops that contain approximately half the total number of vertices. These six loops cannot be seen
as several clusters, but dominate the whole topology; they form a cluster on their own. In each decomposition of the Alarm network, the relevant net contains this cluster. Therefore, the complexity of the relevant net equals the complexity of the original net. In addition, the size of the relevant net, in any case, extends half of the number of vertices in the original Bayesian belief network.

Applying the proposed decomposition method has also been studied in decision-support systems based on Bayesian belief networks. The objective of these decision-support systems is to suggest actions. The process of determining which action to suggest is called the planning of actions. The Bayesian belief network formalism need to be extended in order to enable the planning of actions.

In the literature, several extensions of the Bayesian belief network formalism have been proposed. In this thesis, the extension called decision networks is considered. In a decision network, information is taken into account immediately as it becomes available. In doing so, the planning of action is dedicated to the specific decision problem at hand.

A decision network is a Bayesian belief network where the vertex set is extended by decision vertices and utility vertices. A decision vertex represents actions that can be taken. A utility vertex represents the consequences of actions taken. These consequences are weighted by utilities. During the planning process, for each action, a trade-off is made between the (expected) negative consequences and the (expected) positive consequences. This trade-off is expressed in the utility measure. The decision network suggests the action for which the value of the utility measure is maximal.

In order to calculate the utility measure associated with an action, the marginal probability distributions of the variables corresponding to the utility vertices are required. These probability distributions are calculated in the decision network by propagating the impact of the action throughout the decision network. In general, during this propagation more calculations are performed than necessary for the determination of the required probability distributions. It is demonstrated that the proposed decomposition method is directly applicable in the context of decision networks. Further, a comparative study is conducted in order to place the decision network formalism in the taxonomy of methods for the planning of actions.

In conclusion, it is stated that decision networks are very suitable to be applied in domains where it is essential to tune the planning of actions to the constantly
changing information. In addition, the computational intensity required for this flexibility can be kept under control by applying the proposed decomposition method.
Chapter 1

Introduction

The term Artificial Intelligence (AI) is used to denote a wide range of research areas. These include diagnostic, predictive and abductive reasoning, planning, robotics with associated perception (vision), symbolic manipulations in mathematics, theorem proving, game playing, natural language processing, and so on. The areas of artificial intelligence research mentioned above have in common that the research is directed toward the development of systems that are capable of performing tasks originally done by human beings (experts) only. These systems should show some sort of intelligent behavior, of which common-sense reasoning is one important aspect. To this end, these systems incorporate and manipulate knowledge which gave their name, knowledge-based systems.

It has largely been acknowledged that uncertainty plays a key role in today’s knowledge-based systems. The need to represent uncertainty appears because of various aspects of the knowledge to be represented. Concerning this representation, we mention the incompleteness of the knowledge and the vagueness of the descriptors in the language used to communicate about the domain to be represented. In addition, the inputs to trigger the reasoning may be imprecise, unreliable, vague and incomplete. These various aspects of uncertainty have given rise to numerous manners of handling uncertainty.

In the following, we give a brief overview of the numerical approaches to reasoning with uncertainty. By no means do we aim at completeness, instead, the emphasis is on placing the (computational complexity of) probabilistic inference in Bayesian belief networks in the context of other well-known approaches to reason with uncertainty.
1.1 Early approaches

The early numerical approaches toward the incorporation of uncertainty can be divided into the rule-based and model-based. Model-based systems incorporate a probabilistic model, that is a joint probability distribution of the variables discerned, together with methods to manipulate this joint probability distribution. In these model-based systems, probabilistic consistency of the inference is guaranteed. This means that, for each variable in the model, the updated probability distribution obtained by (probabilistic) inference conducted in the model is identical to the updated probability distribution obtained after appropriate conditioning and marginalizations of the joint probability distribution.

1.1.1 Idiot Bayesian approach

The early Bayesian approach, also called the Idiot Bayesian approach, belongs to the class of model-based approaches. The Idiot Bayesian approach can be viewed as an attempt to build systems based on pure probabilistic models. The use of the probability theory to handle the inexact reasoning seemed attractive because of its consistency and completeness. However, for a long time, this Bayesian approach has been considered infeasible because, first, a huge number of (conditional) probabilities have to be specified and, second, the number of computations increases exponentially with the number of variables. To overcome these difficulties, simplifying assumptions about the dependencies between the variables in the domain had to be made. In systems that use this approach, see for example [27], two simplifying assumptions are made: the independence of all evidence and mutually exclusive hypotheses. Note that under the second assumption, it becomes impossible to find a diagnosis composed of multiple hypotheses, while the first assumption is not very close to reality.

1.1.2 Rule-based approach

Research on the rule-based approach resulted in quasi-probabilistic models such as the certainty factor model [11] with the well-known application MYCIN [57], developed by Buchanan and Shortliffe [10].

MYCIN is a rule-based system for the diagnosis of patients suffering from a bacterial infection. After the diagnosis has been established, MYCIN assists in finding the optimal antibiotic to administer. To enable reasoning in the presence
of uncertainty, each rule has a certainty factor attached to it. The certainty factors are real numbers with values between \(-1\) and \(+1\), where positive values indicate that the verity of the premise should increase our belief in the conclusion, and negative values indicate that the verity of the premise should decrease our belief in the conclusion. Certainty factors are easy to understand and computationally cheap. Moreover, systems incorporating them perform remarkably well, see for example [75] and, therefore, certainty factors found quick acceptance and are still being frequently used. A collection of the major results from the MYCIN project, and the lessons learned (in retrospect), which apply to rule-based systems and the uncertainty management technique of certainty factors in general, have been published in [11].

However, certainty factors have been subject to analysis and criticism, see for example [67, 25]. These analyses concentrated on establishing an interpretation for a certainty factor of a rule, say, IF E THEN H, which is essential for the assessment of the certainty factor. Heckerman [28] has shown that every probabilistic interpretation of the certainty factor is a transformation of the likelihood ratio, \( \frac{P(E|H)}{P(E|-H)} \). However, the certainty factor model is not a special case of probability theory since it incorporates heuristic combination functions. The inference calculus is equivalent to the one used in the Idiot Bayesian systems, this means that the same conditional independence assumptions are being made as those which are implicitly present in the combination functions.

With a mathematical foundation for the inference process available, analyzing the behavior of rule-based systems with certainty factors would become possible. Van der Gaag [67] derived the independence assumptions that must be made in order to render the propagation functions for the implication and the combination correct, with respect to the probabilistic definition of a certainty factor. Goedhart [25] completed this analysis with his demonstration that a number of conditions and interpretations are necessary in order to make the components of which a certainty factor is composed internally consistent with respect to their probabilistic definition.

Another quasi-probabilistic model of belief functions was developed by Dempster [18] and later extended by Shafer [56]. This Dempster-Shafer model represents uncertainty by a range of probabilities. Gordon [26] showed that this model has a firm mathematical foundation. The model can be considered as an extension of probability theory in the sense that it leaves room to express ignorance. The mathematical foundation seems to be an advantage over the certainty factor model. A
problem with the Dempster-Shafer model is the absence of propagation functions. Various propagation functions have been proposed [56, 19]. However, if these functions are chosen to represent real interactions, the computational complexity of the belief functions becomes combinatorial.

Opposed to probabilistic approaches for representing uncertainty are the possibilistic approaches with as their founder L. A. Zadeh [76, 77]. In possibilistic systems, the type of uncertainty represented is vagueness and imprecision as is present in natural language. In comparison with probabilistic approaches, the uncertainty is present in the description of events rather than in the occurrence of events.

Possibilistic reasoning approaches are based on fuzzy sets and many-valued fuzzy logics. A variable can take a number of values simultaneously, albeit with different degrees of satisfiability. A heated debate is going on among advocates of the probabilistic and those of the possibilistic approach, about the necessity of introducing other measures of uncertainty than probabilistic ones. See for example [12] and [78] for two opposite views. In this thesis, we do not enter into the debate about the appropriateness of probability theory to handle uncertainty in reasoning systems.

Nevertheless, an attempt to present only one knowledge representation form and one inference mechanism as being superior to all others seems narrow-minded. It is more likely that future reasoning systems that have to deal with large domains and diverse tasks will be of the hybrid type, combining a selection of formalisms from the full range available, each best suited for a part of the domain and a specific task. For example, property description could be represented in terms of fuzzy predicates, heuristic expert knowledge could be represented as a rule base, a classification task could be carried out by a neural network, for abductive reasoning and the planning of actions, a Bayesian belief network could be used.

To conclude we make the following remark concerning the functionality of rule-based systems. It has been recognized that the inference mechanism of logical deduction can be applied only if the principle of modularity [29, 30] can fully be realized in the application domain. In a system that uses deductive logic as the inference mechanism, the modularity principle holds. We note that the modularity of deductive reasoning does not depend on whether or not uncertainty is present. Moreover, if a consistent deductive reasoning system is augmented with uncertainty, the underlying logic prevents the derivation of contradictions. On the one hand, this explains why deductive rule based systems augmented with
(un)certainty measures perform satisfactorily. But, on the other hand, only a weighing of the derived conclusions is achieved after the addition of uncertainty. It is argued that, for this reason, deductive inference alone is not powerful enough to obtain plausible human-like reasoning, and more flexible inference mechanisms are needed for this.

1.2 Extensional and intensional

A taxonomy of reasoning systems based on the certainty calculus they incorporate does not provide a good classification of the behavior of the systems. For example, probability theory can be used in a rule-based setting, as in PROSPECTOR, as well as in a model-based setting, as in MUNIN [3].

Another grouping of reasoning systems according to the treatment of uncertainty provides a more fundamental difference. The distinction in this taxonomy amounts to the difference between extensional and intensional systems [47].

Extensional systems are based on logic theory and treat uncertainty as a generalized truth value. A rule based system is a typical example of an extensional system. The certainty of the premise set of a rule summarizes the past actions of the problem solver, while the conclusion explicitly describes the next action. When a sequence of rules does not lead to the desired result, it is common practice to invoke a backtracking mechanism to gather information disregarded at an earlier stage. In the case of multiple choices for rule selection, the problem solver generates multiple solutions called extensions. We note that, in general, the correlation between the certainties of the extensions is not accounted for, and the determination of an ordering of the extensions with respect to their certainty needs an additional procedure.

Intensional systems are based on set theory and assign certainty measures for sets of possible worlds. Rules in these systems should be interpreted as relations between the variables involved which constrain the possible worlds. The inference mechanism transfers the belief status of the present world into a new state consistent with the whole set of observed evidence. Backtracking is not needed because, at any instant in time, the next state of the world is determined with knowledge of all relevant information. The Dempster-Shafer formalism, the Idiot Bayesian approach, and the Bayesian belief network models are examples of intensional systems.
1.3 Bayesian belief networks

A major breakthrough in the research on sound probabilistic models was the development of Bayesian belief network models. From a memory requirement point of view, a Bayesian belief network is an efficient representation of the joint probability distribution of the variables discerned in the domain. The technique to represent a joint probability distribution in a factorized form by using graphs was already around before the introduction of Bayesian belief networks, see for example [39]. The efficiency of using these graphical representations applied only to memory requirement. The calculation of a marginal probability distribution amounted to the multiplication of all factors followed by the appropriate marginalizations. In a Bayesian belief network, the computational efficiency of determining a marginal probability distribution has been improved, in comparison with the calculation of marginal probability distribution based on the joint probability distribution in non-factorized form. A directed graph is used to represent independence relations that are exploited to this end [46, 42, 53].

Important achievements have been made in the formalization of the connection between the independence relations expressed in the graph and in the joint probability distribution of the variables in the domain [68, 48, 23].

As the field matured, building real-world applications received an increasing amount of attention. Consequently, the construction of network models became an issue [32, 45]. The structure of a Bayesian belief network has been argued to be the predominant factor determining the model’s performance. Getting the structure right demands both modeling skills and an understanding of the application domain. Each modeling task is more or less unique, and at present, generally applicable tools to assist the construction of Bayesian belief network models are not available.

Concerning the numerical part, the assessment of the conditional probabilities can be simplified by the use of interaction models. The research on interaction models resulted in, for instance, the noisy or-gate model [47] and the extended linear models [2].

1.3.1 Singly connected networks

One of the first network models was proposed by J. H. Kim and J. Pearl [40]. The network possesses a directed singly connected structure, which enables a very efficient algorithm for probability updating in which the tree structure is
1.3 Bayesian belief networks

used as computational architecture. This means that the calculations involved in the updating of probabilities, the propagation process, are local to the vertices in the network. For an individual vertex, only the direct neighborhood of parent and child vertices is relevant. The impact of new information spreads through the network by local message passing between the vertices. The message-passing process is guaranteed to stop resulting in, for each variable, an updated probability distribution probabilistically consistent with the findings.

The messages can be given an intuitive interpretation, they indicate either diagnostic or predictive support for the receiving vertex, depending on the direction of the arc over which they are sent. Further, according to Pearl, the reasoning patterns that emerge are plausible. The basis for the patterns are the independence relations embedded in the network model. In essence, the dependency structure of the network dictates which information is relevant with respect to an individual inference step, and which information need not to be considered. These relevance relations are sensitive to the dynamically changing context of the available evidence. The combination of structural and probabilistic knowledge reduces the amount of data as well as the number of necessary numerical computations.

The strengths of the (causal) relationships in the network are specified by conditional probability tables that represent the degree of belief attached to an event under different conditions. In this thesis, we do not question the validity of the numerical part of a Bayesian belief network. We mention that an analysis of the sensitivity to variation in the probabilities of Bayesian classification models shows the robustness of these models [7].

Compared to the early Bayesian approach, a reduction of the computational complexity has been achieved because independence relations are made explicit and only the restricted tree topologies are allowed. Generalizing the network topology, J. Pearl developed a method for probability propagation for singly connected directed networks. A collection of this work was published in [47]. In this book, methods to manage multiply connected Bayesian belief networks are discussed too. These methods are based on a transformation of the original network into a singly connected computationally equivalent network to which his propagation algorithm can be applied directly.
1.3.2 Multiply connected networks

A method for probability updating in *general* directed acyclic networks was developed by Lauritzen and Spiegelhalter [42]. Contrary to Pearl's approach, plausible reasoning patterns were no issue. Instead, the development of methods to efficiently update the probability distribution of decomposable models was the purpose of the research. This approach was inspired by the existing statistical theory of Markov's random fields, see for example [16, 39].

Based on the work of Lauritzen and Spiegelhalter, a probabilistic inference mechanism was developed [36, 37, 44, 35, 38]. This mechanism has been implemented in the Hugin-Shell [34], which is currently one of the few commercially available reasoning system shells for the development of Bayesian belief networks.

In this method, the original network is transformed into an undirected tree structure, called a *junction tree*. Probability propagation is performed in this junction tree. The calculations involved are local to the vertices of the junction tree. Messages are passed between them to propagate the impact of new information. An interpretation of these messages in terms of predictive or diagnostic support, as in Pearl’s method, is not straightforward, because the messages can represent the combination of diagnostic and causal effects that sets of variables have on each other. Further, because of the multiple connectivity, relevance is not longer restricted to the direct neighbors; the inference mechanism has to keep track of common causes and common consequences.

1.3.3 Computational complexity

With regard to the Bayesian belief network formalism, G. F. Cooper [14] pointed out that it is very unlikely that efficient algorithms will be found that perform probabilistic inference in general; he showed that, in general, probabilistic inference is NP-hard [14]. NP-hard problems belong to the class of most difficult to solve problems, and are characterized by an exponential computational complexity of finding a solution. For a complete treatment of problem classification according to computational complexity we refer to [4].

It is important to note that the use of fast(er) multiple computing devices would not allow the computational problem to evaporate. Only polynomial problems benefit from parallelism in the sense that the order of complexity can be reduced when the number of computing devices equals at least the input size. For exponential problems, it would not suffice to have a number of computing devices
that equals the input size; the problem would remain exponential. This can be seen if we realize that the number of alternative computations to be carried out grows with the number of executed computations.

In respect to the previous discussion, we conclude that the computational complexity of probabilistic inference in Bayesian belief networks poses a severe problem in their practical use in large application domains. The implications of the complexity analysis must be taken as a reason to develop special-purpose propagation methods suitable for well-defined Bayesian belief network classes and problem instances.

1.4 Concluding remarks

We do not enter into the discussion as to whether probabilistic inference combined with intensional models naturally results in plausible reasoning patterns. We note that some reasoning patterns in Bayesian belief networks are rarely part of the inference mechanism in rule based systems. We refer to [24] for an exposition of remedies for the imperfections of extensional models in a rule based setting. The presented remedies allow for intercausal reasoning patterns such as explain away and joint explain. Further, the dependence between propositions based on common causes is accounted for. However, the lack of bidirectional inference results in only partially consistent explanatory dependency networks.

1.5 Research objectives

The main research objective was the handling of the computational complexity inherent to probabilistic inference in general Bayesian belief networks. To this end, a problem-directed decomposition method has been developed in order to restrict the probabilistic inference to the area of interest in the domain model with respect to a specified problem instance. In doing so, the global probabilistic consistency property is lost, because the variables in the part of the network outside the current area of interest receive no updates for their probability distributions. However, probabilistic consistency within the current area of interest is guaranteed. Moreover, after a change in the problem instance, and accompanying change in the area of interest, probabilistic consistency with the previous area of interest is restored such that the updated probability distributions of the variables in the
current area of interest are identical to the updated probability distributions of these variables as calculated in the complete domain model.

Although probabilistic consistency is a property of the quantitative or numerical part of the Bayesian belief network, the area of interest is inferred from the qualitative or graphical part. As a consequence, the proposed decomposition method is independent of the quantification of the dependency relations. Even if the uncertainties are expressed in a different manner, the procedure to determine the area of interest still yields valid areas of interest, provided that the inference mechanism respects the independence relations as reflected in the directed acyclic graph.

Secondary to the computational complexity issue, the planning of actions in extended Bayesian belief networks, called decision networks, has been studied. Modeling issues have arisen in these studies. In particular, the potentials of the separate representation of domain knowledge and knowledge about strategies has been researched. Of course, the computational complexity problem surfaces again, and the applicability of the decomposition method in this context has been researched.

1.6 Scope of the thesis

The Bayesian belief network formalism is discussed in Chapter 2 and in Chapter 3.

In Chapter 2 we review the established definitions and theorems which are directly relevant to the research presented in the thesis. Most important, the connection between the independence relations expressed in the graph and in the joint probability distribution of the variables in the domain is discussed.

Having established the theoretical background with respect to the structural part of a Bayesian belief network, we give a formal discussion of the inference procedures in Chapter 3.

To cope with the computational complexity of inference in general Bayesian belief networks, a problem-directed decomposition method is proposed in Chapter 4. This method amounts to a transformation of the original Bayesian belief network into a relevant network and a rest network. The dependency structure of the relevant network is determined with respect to a problem instance. A problem instance consists of variables of interest and variables the values of which are known, and is assumed to be associated with a reasoning task. The transformation process acts as a focus of attention on the part of the Bayesian belief network
relevant to the update of the variables of interest. During a reasoning session, the specification of the problem instance may change. The decomposition method is flexible in the sense that changes can easily be accounted for.

To promote insight into the impact of problem-directed decomposition, Chapter 5 reports on an experimental evaluation with both artificially generated inputs and realistic ones. The results of the evaluation provide insight into the impact the problem directed decomposition method has on the computational complexity of the propagation process. However, the problems generally present in the evaluation of methods to control the computational complexity of the propagation, where the topology of the network play a dominant role, become clear.

The planning of actions is the subject of Chapter 6. The probabilistic nature of (un)certainty with which causal relations affect the variables involved enables the straightforward extension of a Bayesian belief network to enable decision analysis in order to plan actions. The potentials of extended Bayesian belief networks, called decision networks, are discussed in relation to traditional decision analysis. The advantages of decision networks become clear, but so does the disadvantage: the computational complexity. The traditional trade-off between representation sparsity and computational complexity surfaces. We show that the problem-directed decomposition method can be used to reduce the computational complexity of the planning task.

In Chapter 7, this thesis is concluded with a discussion, open questions and recommendations for further research.
Introduction
Chapter 2

Graphical models

In the introduction, it is made clear that the uncertainty present in a knowledge domain is represented probabilistically in the associated Bayesian belief network, and that the reasoning in a Bayesian belief network amounts to the calculation of the marginal probability distributions of the variables discerned in the domain given the evidence. The reasoning in Bayesian belief networks derives its power from the faithful representation of dynamically changing dependency relations among the variables discerned in the knowledge domain of interest. The dependency relations are represented by the structure of the Bayesian belief network. The structure of the network together with the known facts determine the reasoning patterns.

In this chapter, we review the results on the research of graphical representations of probabilistic models [48, 21, 68]. The emphasis is on the mapping of (conditional) independence relations as present in the joint probability distribution of the variables discerned in the knowledge domain, that is the mapping of the probabilistic dependency model to a graphical structure.

The suitable graphical structure is the directed acyclic graph. Independence relations are read off the graph using a graph-separation criterion adapted to cope with the (extended) expressiveness of a directed representation. The criterion is called \textit{d}(irected)-\textit{s}eparation. After acknowledgment of the representation form, understanding the d-separation criterion is the objective, because of the key role it plays in both the mapping of the probabilistic dependency model on the directed acyclic graph, and during probabilistic inference. It is demonstrated that, if we take the directions of the arcs in the graph to represent the notion of \textit{causality}, the d-separation criterion yields intuitive independence relations among the variables
concerned.

The remainder of this chapter is organized as follows: Section 2.1 provides the general concept of a dependency model. In Section 2.2, the special type of dependency model characterized by the property that it can be represented by an undirected graph is discussed. In Section 2.3, the directed graphical representation is discussed together with the d-separation criterion. The directed representation is able to capture more (in)dependence relations present in a joint probability distribution than the undirected representation, but it is still not able to capture both all dependence relations and all independence relations. Approximate mappings are unavoidable and are also discussed. This chapter is concluded by a discussion in Section 2.4.

2.1 Dependency models and independency statements

An independency model $M$ defined on a set of propositional variables $U$ is a set of independency statements $I(X, Z, Y)_M$, where $X, Y, Z$ are tree disjoint subsets of $U$. The independency statement $I(X, Z, Y)_M$ asserts that, in $M$, the interaction between $X$ and $Y$ is controlled by $Z$. In particular, $Z$ forces $X$ and $Y$ to be non-interaction subsets of $U$.

2.2 Dependency models characterized by undirected graphs

Any independency model $M$ can be represented by enumerating all independency statements. Since this approach requires an exponential amount of space, which in addition renders the extraction of a specific independency statement cumbersome, more efficient representations have been searched for. A graph is such a more efficient representation of an independency model.

We first review the graph theoretic notions that are important in the remainder.

Definition 2.2.1 (Undirected graph) An undirected graph $G$ is an ordered pair $G = (V(G), E(G))$, where $V(G) = \{V_1, V_2, \ldots, V_n\}, n \geq 1$, is a finite set
of vertices or nodes and \( E(G) \) is a finite set of unordered pairs \((V_i, V_j)\), where \( V_i, V_j \in V(G) \) and \((V_i, V_j)\) denotes an edge between \( V_i \) and \( V_j \).

\( \text{Adj}_G(V_i) = \{ V_j : (V_i, V_j) \in E(G) \} \) denotes the set of vertices adjacent to \( V_i \) in \( G \).

**Definition 2.2.2** (Chain) Let \( G = (V(G), E(G)) \) be an undirected graph.

A sequence of vertices \([V_0, V_1, \ldots, V_m]\) is a chain of length \( m \) in \( G \) between \( V_0 \) and \( V_m \) if \((V_{i-1}, V_i) \in E(G)\) for \( i = 1, 2, \ldots, m \).

**Definition 2.2.3** (Connected undirected graph) Let \( G = (V(G), E(G)) \) be an undirected graph.

A graph \( G = (V(G), R(G)) \) is connected if there is a chain \([V_i, \ldots, V_j]\) in \( G \) for any two distinct vertices \( V_i, V_j \in V(G) \).

**Definition 2.2.4** (Simple chain) Let \( G = (V(G), E(G)) \) be an undirected graph.

A chain \([V_0, V_1, \ldots, V_m]\) is simple if \( V_i \neq V_j \) for \( i \neq j \) and \( 0 \leq i \leq j \leq m \).

**Definition 2.2.5** (Cycle) Let \( G = (V(G), E(G)) \) be an undirected graph.

A cycle is a chain \([V_0, V_1, \ldots, V_{m-1}, V_0]\), such that the \([V_0, V_1, \ldots, V_{m-1}]\) is a chain of at least length 2. A cycle is simple if \([V_0, V_1, \ldots, V_{m-1}]\) is a simple chain.

**Definition 2.2.6** (Simple graph) Let \( G = (V(G), E(G)) \) be an undirected graph.

The graph \( G \) is simple if \( E(G) \) is a set such that \((V_i, V_i) \notin E \) for all \( V_i \in V(G) \).

In the sequel, we consider only simple graphs, undirected or directed.

**Definition 2.2.7** (Graph separation in undirected graphs) Let \( G = (V(G), E(G)) \) be an undirected graph. And let \( X, Y, Z \) be three disjunct subsets of \( V(G) \).

\( X \) and \( Y \) are separated in \( G \) by \( Z \) if and only if any path from a vertex in \( X \) to a vertex in \( Y \) contain at least one vertex in \( Z \). Notation:

\([X \mid Z \mid Y]_G\)

An independency model \( \mathcal{M} \) defined on the set of variables \( U \) can be represented by an undirected graph \( G = (V(G), E(G)) \), in which there is a direct
correspondence between the set of vertices $V(G)$ and the elements of $U$, such that the independency statements in $\mathcal{M}$ correspond with graph separation in $G$.

In the ideal case, the graph topology reflects both all independence assertions and all dependence assertions captured in $\mathcal{M}$, i.e.

\[(X \mid Z \mid Y)_G \implies I(X, Z, Y)_{\mathcal{M}} \text{ and } (X \mid Z \mid Y)_G \iff I(X, Z, Y)_{\mathcal{M}}\]

This correspondence is called graph isomorphism and provides a means to represent every independence assertion as well as every dependence assertion captured in $\mathcal{M}$.

A problem that arises when undirected graphs are used to represent independency models is the deficiency to enable intercausal influence among causes with possess a common consequence [47, 61]. These induced dependencies relations cannot be represented by undirected graphs because in every undirected graph $G$ holds:

\[(X \mid Z \mid Y)_G \implies (X \mid Z \cup W \mid Y)_G.\]

If $X$ and $Y$ are separated by $Z$ in $G$, then they will always remain separated after adding an arbitrary subset $W$ to the separation set $Z$.

### 2.3 Dependency models characterized by directed acyclic graphs

From the previous, we know that the expressiveness of undirected graphs is too limited to represent induced dependencies. Fortunately, induced dependencies can be expressed in the richer language of directed acyclic graphs.

Analogous to the previous section we start with the introduction of the preliminary graph theoretic notions.

**Definition 2.3.1 (Directed graph)** A directed graph $G$ is an ordered pair $G = (V(G), A(G))$, where $V(G) = \{V_1, V_2, \ldots, V_n\}, n \geq 1$, is a finite set of vertices or nodes and $A(G)$ is a finite set of ordered pairs $(V_i, V_j)$, where $V_i, V_j \in V(G)$ and $(V_i, V_j)$ denotes an arc from $V_i$ pointing to $V_j$.

The vertices $V_i$ and $V_j$ are adjacent and $V_i$ is a parent of $V_j$ and $V_j$ is a child of $V_i$.

$s_G(V_i) = \{V_j : (V_i, V_j) \in A(G)\}$ denotes the set of children of $V_i$ in $G$. $p_G(V_i) = \{V_j : (V_j, V_i) \in A(G)\}$ denotes the set of parents of $V_i$ in $G$. The subscript $G$ will be omitted when the graph concerned is clear from the context.
Definition 2.3.2 (Chain) Let $G = (V(G), A(G))$ be a directed graph.
A sequence of vertices $[V_0, V_1, \ldots, V_m]$ is a chain of length $m$ in $G$ between $V_0$ and $V_m$ if $(V_{i-1}, V_i) \in A(G)$ or $(V_i, V_{i-1}) \in A(G)$ for $i = 1, 2, \ldots, m$.

Definition 2.3.3 (Connected directed graph) Let $G = (V(G), A(G))$ be a directed graph.
A graph $G = (V(G), A(G))$ is connected if there is a chain $[V_i, \ldots, V_j]$ in $G$ between any two distinct vertices $V_i, V_j \in V(G)$.

Definition 2.3.4 (Simple chain) Let $G = (V(G), A(G))$ be a directed graph.
A chain $[V_0, V_1, \ldots, V_m]$ is simple if $V_i \neq V_j$ for $i \neq j$ and $0 \leq i \leq j \leq m$.

Definition 2.3.5 (Path) Let $G = (V(G), A(G))$ be a directed graph.
A sequence of vertices $[V_0, V_1, \ldots, V_m]$ is a path of length $m$ in $G$ from $V_0$ to $V_m$ if $(V_{i-1}, V_i) \in A(G)$ for $i = 1, \ldots, m$.
A path $[V_0, V_1, \ldots, V_m]$ is simple if $V_i \neq V_j$ for $i \neq j$ and $0 \leq i \leq j \leq m$.

Definition 2.3.6 (Cycle) Let $G = (V(G), A(G))$ be a directed graph.
A cycle is a path $[V_0, V_1, \ldots, V_{m-1}, V_0]$, such that the $[V_0, V_1, \ldots, V_{m-1}]$ is a path of at least length 2. A cycle is simple if $[V_0, V_1, \ldots, V_{m-1}]$ is a simple path.

Definition 2.3.7 (Directed acyclic graph) Let $G = (V(G), A(G))$ be an directed graph. Then $G$ is acyclic if $G$ contains no cycles.

Next, we firstly discuss the graph separation criterion in directed acyclic graphs, called d-separation. Secondly, we discuss the mapping of a probabilistic dependency model onto a directed acyclic graph.

2.3.1 d-separation

We discuss the d-separation criterion by using typical examples of directed acyclic graph topologies. We discuss the independence relations which intuitively hold if the direction of the arcs is taken to represent the notion of causality. After that, we proceed with the definition of d-separation. Finally, in order to show that the intuitive independence relations found in the informal discussion can be derived from the directed acyclic graphs by using the d-separation criterion, we discuss the examples again, but now apply the d-separation criterion and show that identical independence assertions are derived.
Consider the directed acyclic graph $G$ shown in Figure 2.1.a. $\{A\}$ is the only cause of $\{B\}$ and $\{C\}$. When we substitute cold for $\{A\}$, fever for $\{B\}$ and sore throat for $\{C\}$ we obtain an example of this type of network.

If we do not know if a cold is present, knowledge about the fever would give some prediction of the presence of sore throat. Thus, we would expect fever and sore throat to be dependent. However, once we know that the only possible cause cold is present, the values of fever and sore throat will be determined completely. Changes in the value of fever can tell us nothing more about the value of sore throat and vice versa. These observations imply the presence of an independency model $\mathcal{M}$ such that $I(\{B\}, \emptyset, \{C\})_G \notin \mathcal{M}$ and $I(\{B\}, \{A\}, \{C\})_G \notin \mathcal{M}$.

Consider the directed acyclic graph, $G$, of Figure 2.1.b. An example of this type of network is obtained by substitution of inadequately dressed for $\{B\}$, cold for $\{A\}$ and fever for $\{C\}$. From the notion of causality, we would expect that inadequately dressed tells us something about the fever, since inadequately dressed causes cold and cold causes fever. However, if we know a cold is present, that is if we know the value of cold, knowledge about how the cold was caught, that is knowledge about the value of inadequately dressed no longer tells us anything about the value of fever. These observations imply the presence of an independency model $\mathcal{M}$ such that $I(\{B\}, \emptyset, \{C\})_G \notin \mathcal{M}$ and $I(\{B\}, \{A\}, \{C\})_G \notin \mathcal{M}$. Note that these two independence statements are identical to the previous ones.

Consider the directed acyclic graph, $G$, shown in Figure 2.1.c. We substitute cold for $\{B\}$, pneumonia for $\{C\}$ and fever for $\{A\}$. Both cold and pneumonia are independent explanations for fever. However, once we observe the common consequence fever; the causes cold and pneumonia become dependent, for example, as competing explanations for fever. These observations imply the presence of an independency model $\mathcal{M}$ such that $I(\{B\}, \emptyset, \{C\})_G \in \mathcal{M}$ and $I(\{B\}, \{A\}, \{C\})_G \notin \mathcal{M}$. In this case, $\{B\}$ and $\{C\}$ are induced dependent.

Using the d-separation criterion, given in the next definition, we are able to derive these independence relations from the structure of $G$.

![Figure 2.1: Typical topologies.](image)

**Definition 2.3.8 (d-separation)** Let $G = (V(G), A(G))$ be a directed acyclic graph with vertex set $V(G) = \{V_1, \ldots, V_n\}$ and arcs $A(G)$. Let $X, Z, Y$ be three
disjoint subsets of vertices in $G$. A set of vertices $Z$ d-separates the two sets $X$ and $Y$ if and only if every path from a vertex in $X$ to a vertex in $Y$ is blocked by $Z$.

**Definition 2.3.9 (Blocked path)** Let $G = (V(G), A(G))$ be a directed acyclic graph with vertex set $V(G) = \{V_1, \ldots, V_n\}$ and arcs $A(G)$. Let $Z$ be a subset of vertices in $G$ and let $V_i, V_j \in V(G) \setminus Z$.

In $G$, a path $p$ between $V_i$ and $V_j$ is blocked by $Z$ if and only if at least one of the following conditions is true.

1. $\ldots \leftarrow V_k \rightarrow \ldots$ and $V_k \in Z$;
2. $\ldots \leftarrow V_k \rightarrow \ldots$ and $V_k \in Z$;
3. $\ldots \leftarrow V_k \rightarrow \ldots$ and $V_k \in Z$;
4. $\ldots \leftarrow V_k \rightarrow \ldots$ and $V_k \notin Z$.

Condition 3 and condition 4 involve symmetrical topologies in $G$.

We discuss again the typical examples given in Figure 2.1 to illustrate the d-separation criterion.

Consider the directed acyclic graph, $G$, given in Figure 2.1.a. We already know from an intuitive (causal) interpretation of the $G$ that $I(\{B\}, \emptyset, \{C\}) \notin M$ and $I(\{B\}, \{A\}, \{C\}) \in M$. According to Definition 2.3.9 point (1) we find that $\{B\}$ and $\{C\}$ are d-separated by $\{A\}$ if and only if the value of $\{A\}$ is known.

Consider the directed acyclic graph, $G$, shown in Figure 2.1.b. The intuitive causal interpretations yields: $I(\{B\}, \emptyset, \{C\}) \notin M$ and $I(\{B\}, \{A\}, \{C\}) \in M$. According to Definition 2.3.9 point (2) we find that $\{B\}$ and $\{C\}$ are d-separated by $\{A\}$ if and only if the value of $\{A\}$ is known.

Consider the directed acyclic graph, $G$, shown in Figure 2.1c. The intuitive causal interpretations yields: $I(\{B\}, \emptyset, \{C\}) \in M$ and $I(\{B\}, \{A\}, \{C\}) \notin M$. According to Definition 2.3.9 point (4), we find that $\{B\}$ and $\{C\}$ are d-separated if and only if the value of $\{A\}$ is unknown.
2.3.2 Directed acyclic graph representations of probabilistic dependency models

Completely analogous to undirected graph isomorphism, a dependency model is directed acyclic graph isomorphic if and only if

\[(X \mid Z \mid Y)_G \Rightarrow I(X, Z, Y)_P \text{ and } (X \mid Z \mid Y)_G \Leftarrow I(X, Z, Y)_P.\]

where $G$ is a directed acyclic graph. Unfortunately, there are many probabilistic dependency models for which, even in a directed representation, no directed acyclic graph isomorphic representation exists. In those cases, we have to settle for a graphical representation that is an approximation of the dependency model.

Approximating graphical representations for a dependency model either display dependence relations or independence relations faithfully. These graphical representations are called D-maps and I-maps, respectively.

**Definition 2.3.10 (I-maps, D-maps, and Perfect maps)** Let $G = (V(G), A(G))$ be a directed acyclic graph. Let $\mathcal{M}$ be a dependency model for the set of elements $U$.

The graph $G$ is called a dependency map or D-map of $\mathcal{M}$ if for all disjoint $X, Y, Z \subseteq U$:

\[I(X, Z, Y)_\mathcal{M} \Rightarrow (X \mid Z \mid Y)_G\]  \hspace{1cm} (2.1)

The graph $G$ is called an independency map or I-map of $\mathcal{M}$ if for all disjoint $X, Y, Z \subseteq U$:

\[I(X, Z, Y)_\mathcal{M} \Leftarrow (X \mid Z \mid Y)_G\]  \hspace{1cm} (2.2)

$G$ is called a perfect map for $\mathcal{M}$ if $G$ is both a D-map and an I-map for $\mathcal{M}$.

In a D-map, adjacent vertices correspond with dependent variables, and non-adjacent vertices can correspond with either dependent or independent variables. In an I-map, nonadjacent vertices correspond with independent variables and adjacent vertices can correspond with either dependent and or independent variables. For example, the trivial D-map is the empty graph and the trivial I-map is the complete graph.
2.4 Discussion

To summarize, we state that the graphical representation of the probabilistic dependency model that is capable of representing induced dependency relations must be directed. We note that, in principle, the topologies represented in Figure 2.1a and 2.1b represent identical dependency models.

Further, the class of probabilistic dependency models is so broad that there are models for which no perfect map exists in directed acyclic graphs. Therefore, approximate mappings that represent as many independence relations as possible are the best one can come up with. Stated differently, in the process of modeling a probabilistic dependency model in a directed acyclic graph, we aim at finding the minimal-arc I-map.

In practice, very often there is no explicit knowledge about the joint probability distribution of the variables discerned in the knowledge domain. The approach taken in this situation is to model the causal dependency relations between the variables discerned in the knowledge domain, without concerning ourselves about constructing the joint probability distribution, but, instead, based on knowledge about the structure of the domain obtained in collaboration with a human domain expert. The joint probability distribution emerges after the vertices have been quantified by either prior or conditional probability distribution [3, 6, 58].

On those occasions where we have a huge data base of domain cases at our disposal, that is the joint probability distribution in the form of a contingency table, methods that learn the structure of the Bayesian belief network which captures as much as possible independence relations can be used to get an initial domain structure [9, 22].
Chapter 3

Bayesian belief networks

3.1 Introduction

In the previous chapter, we discussed the directed acyclic graph as a means to represent as many as possible independence assertions captured in the joint probability distribution $P$ of the variables $U$ discerned in a knowledge domain of interest. Informally, a Bayesian belief network consists of such a directed acyclic graph $G$, where the vertices $V(G)$ (corresponding to $U$) are quantified with (conditional) probability distributions according to the factorization of $P$ with respect to $G$. In this chapter, we discuss the methods for probabilistic inference in singly connected Bayesian belief networks, as proposed by Pearl [40], and in multiply connected Bayesian belief networks developed by Jensen [35, 38]. The latter method is based on the method proposed by Spiegelhalter and Lauritzen [42] and the various discussions of this paper, see for example [36, 44].

The process of probabilistic inference, or propagation process, benefits from the expressed independence assertions because it utilizes the directed acyclic graph topology. The propagation process in singly connected Bayesian belief networks can be described as a message passing process where the arcs of the directed acyclic graph are used as communication channels. The direction in which a message might be sent over a communication channel is not fixed. A message can be sent in the direction of the arc, which is interpreted as predictive or causal reasoning, or in the opposite direction, which is interpreted as diagnostic or evidential reasoning. During the propagation process, both diagnostic and causal reasoning can take place. In particular, the induced dependency relation is established by causal reasoning followed by diagnostic reasoning and resulting in an intercausal
interaction. We recall from the introduction that the realization of induced dependency relations is one of the most powerful properties of inference in Bayesian belief networks, since they occur in the process of abductive reasoning, that is reasoning from the evidence back to the hypotheses that explain the evidence.

In Section 3.2, the formal definition of a Bayesian belief network is provided. The propagation and updating formulas for singly connected networks are presented in Section 3.3.

When we apply Pearl's method in multiply connected networks, one of the problems is that the messages might circulate indefinitely. The fundamental problem, however, is that the impact of a piece of evidence can be distributed along two or more paths. When this happens, information about the common origin of the impacts is lost in the propagation process. As a result, the multiple impacts are combined independently as the paths meet again. This means that crucial information about the relation between the separated impacts is lost. A solution to this problem is to allow the passing of messages that consist also of relations between variables rather than only probabilities on a single variable. This solution is realized in the propagation of probabilities in junction trees. A junction tree is a transformed version of the original network. The transformation is the subject of Section 3.4.1. The propagation process utilizes the junction tree as a computational architecture. Probabilistic inference in junction trees is discussed in Section 3.5.

3.2 Definitions and notations

In this section, we give a formal description of the general notion of a Bayesian belief network. First, we need to introduce some notational conventions.

Definition 3.2.1 (Configuration) Let $U = \{U_1, U_2, \ldots, U_n\}$ be a finite set of propositional variables.

A propositional variable $U_i \in U$ can take one state of a finite set of possible states $\{u_{i,1}, u_{i,2}, \ldots, u_{i,n}\}$. This set is called the state space and denoted by $Sp(U_i) = \{u_{i,1}, u_{i,2}, \ldots, u_{i,n}\}$. In case where the $U_i$ is binary the state space is denoted by $Sp(U_i) = \{u_i, \neg u_i\}$, where $U_i = u_i$ corresponds to $U_i = \text{true}$ and $U_i = \neg u_i$ corresponds to $U_i = \text{false}$.

An instantiation of a variable is the fixation of this variable to one of its states, for example $U_i = u_{i,1}$.
3.2 Definitions and notations

The conjunction \( U_1 \land U_2 \land \ldots \land U_n \) is denoted with \( C_U \). A configuration is obtained from \( C_U \) by replacing each \( U_i \) with a value from its state space. The set of all possible configurations obtained from \( C_U \) is denoted with \( C_U^* \).

Definition 3.2.2 (Bayesian belief network) A Bayesian belief network \( B \) is defined as \( B = (G, \mathcal{P}) \), where \( G = (V(G), A(G)) \) is a directed acyclic graph with vertices \( V(G) = \{V_1, V_2, \ldots, V_n\} \), \( n > 1 \), and arcs \( A(G) \). The arcs, \( A(G) \), represent (causal) relationships among the variables \( V(G) \).

A non-negative conditional probability assessment function is assigned to each vertex \( V_i \): \( \mathcal{P}(V_i \mid C_{pa}(V_i)) \in \mathcal{P} \) such that,

\[
\forall c \in C_{pa}(V_i) \sum_{v_{i,j} \in \mathcal{P}(V_i)} \mathcal{P}(v_{i,j} \mid c) = 1
\]

The initial probability assessment functions of a Bayesian belief network define a joint probability distribution on the variables in \( V(G) \) as is stated in the following proposition [39],

Proposition 3.2.1 Let \( B = (G, \mathcal{P}) \) be a Bayesian belief network.

The set \( \mathcal{P} \) defines the joint probability distribution \( P \) on the set \( V(G) = \{V_1, \ldots, V_n\} \) which obeys the independence relations as presented in \( G \) as follows:

\[
P(C_{V(G)}) = \prod_{V_i \in V(G)} \mathcal{P}(V_i \mid C_{pa}(V_i))
\]

Thus \( P \) is the joint probability distribution on the variables \( V(G) \) such that \( G \) is an I-map of \( P \).

To illustrate the notion of a Bayesian belief network consider the Bayesian belief network depicted in Figure 3.1. The directed acyclic graph topology reflects conditional independence relationships among the probabilistic variables \( V(G) \). We have: \( V_7 \) is conditionally independent of \( V_2, V_1, V_4, V_6 \) given \( V_3 \) and \( V_5 \). In terms of probabilities:

\[
P(V_7 \mid V_1 \land V_2 \land V_3 \land V_4 \land V_5 \land V_6) = P(V_7 \mid V_3 \land V_6).
\]

Exploring all independence relationships in the directed acyclic graph we can express the joint probability distribution \( P \) as follows.

\[
P(V_1 \land V_2 \land \ldots \land V_7) = P(V_3 \mid V_1 \land V_2) \times P(V_6 \mid V_3) \times P(V_7 \mid V_3 \land V_5) \times P(V_4 \mid V_2) \times P(V_4 \mid V_1) \times P(V_2)
\]
\[ P = \{ P(v_1), P(v_2), P(v_3 \mid v_1 \land v_2), P(v_3 \mid v_1 \land \neg v_2), P(v_3 \mid \neg v_1 \land v_2), P(v_3 \mid \neg v_1 \land \neg v_2), P(v_4 \mid v_2), P(v_4 \mid \neg v_2), P(v_5 \mid v_4), P(v_5 \mid \neg v_4), P(v_6 \mid v_3), P(v_6 \mid \neg v_3), P(v_7 \mid v_3 \land v_5), P(v_7 \mid v_3 \land \neg v_5), P(v_7 \mid \neg v_3 \land v_5), P(v_7 \mid \neg v_3 \land \neg v_5) \} \]

Figure 3.1: A Bayesian belief network.

Since the (conditional) probability distributions on the right hand side of the above equation correspond to the probability assessment functions, the joint probability distribution can also be calculated as follows.

\[
P(V_1 \land V_2 \land \ldots \land V_7) = P(V_3 \mid V_1 \land V_2) \times P(V_5 \mid V_6) \times P(V_7 \mid V_3 \land V_5) \times P(V_4 \mid V_2) \times P(V_4) \times P(V_2)
\]

### 3.3 Probabilistic inference in singly connected Bayesian belief networks

In this section, we discuss the method for probabilistic inference in singly connected networks developed by Pearl [47]. The derivation of the presented formulas using the same notational conventions can be found in [61].

**Definition 3.3.1 (Singly connected)** A directed acyclic graph \( G = (V(G), A(G)) \) is singly connected if for every \( U \in V(G) \) and \( V \in V(G) \) there is at most one chain between \( U \) and \( V \).

The single connectivity of the network allows the calculation of the marginal probability distributions of a vertex based on the information passed to this vertex by its direct neighbors only.
Updating

We consider the directed acyclic graph $G = (V(G), A(G))$. Further, we consider an arbitrary vertex $V \in V(G)$, with parent set $\rho_G(V) = \{U_1, \ldots, U_n\}$ and children set $\sigma_G(V) = \{Y_1, \ldots, Y_m\}$. Using Bayes' Theorem and the independence relations represented in $G$ the belief of a variable $V$, $Bel(V)$, is calculated as follows.

$$ Bel(V) = \alpha \times \lambda(V) \times \pi(V) $$(3.1)

where $\alpha$ denotes a normalization constant.

$Bel(V)$ is the probability distribution of variable $V$, updated in the light of the available evidence.

$\lambda(V)$ is calculated from the messages $V$ receives from its children and $\pi(V)$ is calculated from the messages $V$ receives from its parents.

After vertex $V$ has received the messages from its parents, it calculates $\pi(V)$ as follows,

$$ \pi(V) = \sum_{c \in \mathcal{O}_{\pi(V)}} (\mathcal{P}(V | c) \prod_{u_{j,i} \in \mathcal{C}c} \pi_V (u_{j,i})) $$ (3.2)

where $\pi_V (U_j) = (u_{j,1}, u_{j,2}, \ldots, u_{j,k})$ represents the message parent $U_j$ sends to $V$ and $\pi_V (u_{j,i})$ is the $i$-th element of $\pi_V (U_j)$. After vertex $V$ has received the messages from its children, it calculates $\lambda(V)$ as follows,

$$ \lambda(V) = \prod_{Y_i \in \sigma_G(V)} \lambda_{Y_i}(V) $$ (3.3)

where $\lambda_{Y_i}(V)$ represents the message vector child $Y_i$ sends to $V$.

Propagation

After vertex $V$ has received a message from at least one of its neighbors, vertex $V$ generates messages to be sent to the other neighbors. The message $V$ sends to a parent $Y_k$ is calculated as follows.

$$ \pi_{Y_k}(V) = \alpha \times \prod_{Y_i \in \sigma_G(V), i \neq k} \lambda_{Y_i}(V) \times \pi(V) = \frac{\pi(V) \times \lambda(V)}{\lambda_{Y_k}(V)} $$ (3.4)

and the message vertex $V$ sends to a child $U_k$ is calculated as follows.

$$ \lambda_V(U_k) = \alpha \times \sum_{v_i \in \mathcal{S}_V} \left( \lambda(v_i) \left( \sum_{c \in \mathcal{O}_{\pi(V)} \cup U_k} \mathcal{P}(v_i | c \wedge U_k) \prod_{u_{j,i} \in \mathcal{C}c} \pi_V (u_{j,i}) \right) \right) $$ (3.5)
Initialization

Initially, all vertices possess $\lambda$ vectors with a 1 for each possible state. The $\pi$ vectors are calculated by propagating the prior probabilities, specified at the root vertices, through the network.

Input of evidence

Evidence on a variable $V_i$ is information about $P(V_i)$. In the particular case, where this information is a statement that $V_i$ is in a certain state $v_i$, the evidence is called a finding.

A finding is inserted into the network by means of an instantiation of the associated vertex. An instantiated vertex possesses a $\lambda$ vector with value one for the component representing the observed state, and value zero for all other components. Input of evidence at vertex $V_i$ changes the belief in $V_i$ and, as a consequence, initiates the propagation process.

3.4 Probabilistic inference in multiply connected Bayesian belief networks

Recall from the previous sections that Pearl's method for propagating probabilities in singly connected networks utilizes the directed acyclic graph as its computational architecture. Also recall from the introduction that Pearl's method causes problems when applied to multiply connected networks. The message-passing process might not only never stop, because messages circulate indefinitely, but most important, that Pearl's method is not able to coordinate the relations between the decoupled impact of evidence.

Lauritzen and Spiegelhalter proposed a computational scheme for the method for propagating probabilities in multiply connected networks [42]. The method operates on a secondary structure consisting of a collection of sets of nodes organized linearly with the running intersection property. Probabilistic inference is performed in this structure and follows the ordering of the node sets. A substantially simplified propagation scheme was developed by Jensen [35, 38]. In the discussion here, we follow this simplified scheme. The scheme utilizes a derived representation as computational architecture, called a junction tree. In Section 3.4.1, we describe the transformation of an arbitrary Bayesian belief network into a junc-
tion tree representation. The process of probabilistic inference in junction trees is discussed in Section 3.5. Finally, in Section 3.6 an example is discussed to illustrate the propagation process.

### 3.4.1 Junction tree construction

This section is stated with preliminary definitions and notations.

**Definition 3.4.1 (Multiply connected)** A directed acyclic graph \( G = (V(G), A(G)) \) is multiply connected if for at least one pair \( U, V \in V(G) \) there are more than one chains connections them.

Two different chains connection \( U \) and \( V \) are referred to as a loop.

**Definition 3.4.2 (Chord)** In an undirected graph, a simple cycle possesses a chord if there is an arc between two non-consecutive vertices of the cycle.

**Definition 3.4.3 (Complete graph)** An undirected graph \( G = (V(G), E(G)) \), is complete if every pair of distinct vertices is adjacent: \( V_i, V_j \in G(V) : (V_i, V_j) \in E(G) \).

**Definition 3.4.4 (Induced graph)** Let \( G = (V(G), R(G)) \) be a graph (directed or undirected). Given a subset \( W \subseteq V(G) \), the subgraph induced on \( G \) by \( W \) is denoted by \( G_W \). The edges of this subgraph are defined by \( E_W = \{(U, V) \text{ such that } (U, V) \in R \text{ and } U \in W \text{ and } V \in W\} \)

**Definition 3.4.5 (Complete subset)** In an undirected graph \( G = (V(G), E(G)) \), a subset \( W \subseteq V(G) \) is called a complete subset of \( G \) if it induces a complete subgraph.

**Definition 3.4.6 (Clique)** In an undirected graph \( G = (V(G), E(G)) \), a subset \( W \subseteq V(G) \) is called a clique of \( G \) if \( W \) is a complete set and \( W \) is maximal, there is no other clique containing \( W \).

The basic idea underlying the construction of the junction tree for Bayesian belief network \( B = (G, \mathcal{P}) \) is the addition of (undirected) links to \( G \) in order to express explicitly the existence of relations among the probabilistic updates that emerge during the propagation dynamics. Two types of relations are distinguished: induced dependency relations and coordinative dependency relations.
Induced dependency relations are represented by adding a link between any couple of parents that possesses a common child. Recall from Chapter 2 that the only necessity for a directed representation occurs at induced dependencies. To illustrate the junction tree construction, we consider the graph $G$ in Figure 3.2.a. After all induced dependency relations are represented by (added) links, the direction of the arcs can be dropped and we have obtained the moral graph $G_m$, shown in Figure 3.2.b.

**Definition 3.4.7 (Moral graph)** Let $G = (V(G), A(G))$ be an directed acyclic graph. The moral graph, $G_m$, is obtained from $G$ by connecting each pair of vertices with a common child by an undirected edge followed by dropping the direction of the arcs.

The coordinative dependency relations must assure that the impact of evidence is not decoupled and propagated along separated paths to be independently combined again as they meet at some vertex. This is achieved by adding links between variables that are contained in a loop.

Figure 3.2.c depicts a possible result of the addition of the coordinative links. Link $(E, G)$ captures the information that $E$ and $G$ possess a common cause $F$. Knowing the updated probabilities of $E$ and $G$ (with knowledge of $F$) is sufficient to determine the coordinated effects on $H$ and $C$.

Like probabilistic inference in singly connected networks, probabilistic inference in junction trees aims at utilizing the independence relations in order to reduce the computational complexity. The decomposition of the graph in Figure 3.2.c into cliques provides the independence structure which is used as a base for the junction tree.

In fact, the addition of the coordinative links transforms the moral graph $G_m$ into a triangulated graph $G_u$, see Figure 3.2.c.

**Definition 3.4.8 (Triangulated graph)** Let $G = (V(G), E(G))$ be an undirected graph. The graph $G$ is triangulated if and only if every simple cycle of lengths $\geq 4$ possesses a chord.

There are a number of methods to construct a junction tree. In all cases, the tree is based on a triangulated version of the original graph. We note that the triangulation process determines the computational complexity of probabilistic inference and that determining the optimal triangulation is NP-complete [74]. A method for triangulation is discussed in Appendix A.
Based on a triangulated graph, the junction graph is determined.

**Definition 3.4.9 (Junction graph)** A junction graph for an undirected graph $G$ is an undirected labeled graph. The nodes are the cliques in $G$. Every pair of nodes with a non-empty intersection has a link labeled by the intersection.

![Directed acyclic graph](image1)

![Moral graph](image2)

![Triangulated graph](image3)

a: directed acyclic graph  
b: moral graph  
c: triangulated graph

Figure 3.2: Directed acyclic graph of our running example. The numbers between parentheses indicate an ordering obtained by a maximal cardinality search (Appendix A) applied on the moral graph shown in b.

In the junction graph, each clique of $G_u$ is assigned to a node, and two nodes are linked if they possess a non-empty intersection. A link of the junction graph is labeled with the intersection set which is called the separator, and denoted by $S$. The junction graph of a triangulated graph possesses a junction tree. The junction tree can be constructed as the maximum weight spanning tree of the junction graph, where the weight of a link is the number of variables contained in the separator. For a proof of the above statement, we refer to [37]. Figure 3.3.a depicts the junction graph of our running example, and in Figure 3.3.b a possible junction tree is shown. In the junction tree the links connect subsets of variables of $G$. The subset correspond to the cliques in $G_u$. The set of all clique of $G_u$ is denoted by $Cl(G_u)$. In order to complete the junction tree, a clique potential, $\varphi_{Cl_q}$, is attached to each clique. A clique potential (also called potential function or potential table) is calculated as follows.

1. Assign each variable $V_i \in V(G)$ to exactly one clique $Cl_q$, where $V_i \cup \rho(V_i) \subseteq Cl_q$. The assignment of $V_i$ to $Cl_q$ is denoted by $V_i \rightarrow Cl_q$. 
2. The potential function of $Clq_i$ is calculated as the product of the assessment functions of the variables assigned to $Clq_i$ as follows.

$$\varphi_{Clq_i} = \prod_{v_i \in Clq_i} P(v_i | C\varphi(v_i))$$

3. The potential function of separator set $S_i$ is set to be the unity vector.

$$\varphi_{S_i} = \mathbf{1}$$

It should be noted that the logical structure of the knowledge, originally represented by the topology of $G$, remains represented in the clique potentials.

**Definition 3.4.10 (Junction tree)** Let $B = (G, \mathcal{P})$ be a Bayesian belief network as defined in Definition 3.2.2. Further let $G_u$ be a triangulated moral version of $G$. A junction tree of $B$ is defined as $T(B) = (\mathcal{C}, \Phi)$, where $\mathcal{C} = (Cl(G_u), E(G))$ is an undirected tree with nodes $Cl(G_u) = \{Clq_1, Clq_2, \ldots, Clq_n\}$, $n > 1$, and links $E(Cl(G))$. The nodes in $Cl(G)$ correspond to the cliques of $G_u$, and represent subsets of probabilistic variables. The links, $E(Cl(G))$, represent relationships among the cliques $Cl(G_u)$. Each link is labeled with the intersection, $S_i$, of its neighboring nodes. Each vertex $Clq_i$ is quantified by non-negative potential functions, $\varphi_{Clq_i}$. Each separation set is quantified with potential functions, $\mathbf{1}$. $\Phi$ is the set of all potential functions.

We conclude this section with a summary of the junction tree construction procedure.

Let $B = (G, \mathcal{P})$ be a Bayesian belief network. The junction tree $T(B) = (\mathcal{C}, \Phi)$ is constructed as follows,

1. Construct the moral graph $G_m$.
2. Construct the decomposable graph $G_u$.
3. Construct the junction graph.
4. Determine the maximum weight spanning tree $\mathcal{C} = (Cl(G), E(G))$.
5. Assign each variable in $V(G)$ to a clique in $Cl(G_u)$.
6. For each clique $Clq_i$ calculate the clique potential $\varphi_{Clq_i}$, and for each separator set set the potential function to $\mathbf{1}$.
7. $T(B) = (\mathcal{C}, \Phi)$
3.5 Probabilistic inference in junction trees

In this section, we discuss the method for probabilistic inference in junction trees. In the discussion we assume the junction tree to be initialized. That is, the prior probabilities are propagated through the tree giving rise to belief tables. For each clique, Clq, and each separator, Si, the belief tables, B(Clq) and B(Si) respectively, have been calculated. The propagation process is initiated when a piece of evidence is entered into the junction tree. Evidence on a variable V_i is information about the probability distribution of V_i such that P(V_i) changes to P*(V_i).

We start with explaining the basic operation called calibration. A clique U calibrates to a clique V in order to restore consistency among the belief tables. The belief tables of clique U and clique V are consistent if normalization makes them identical. The consistency among clique U and clique V is denoted as follows.

\[ \sum_{U \cap (V \cap W)} B(U) \propto \sum_{V \cap (U \cap W)} B(V) \]

A calibration of U to V consists of two operations, a projection of V to the separation set S, followed by an absorption of U from the separation set S.

Let us assume that B(V) is changed to B*(V) by entering some finding. Clique U calibrates to clique V. Clique V sends the following message to the separator.
\[ S = U \cap V. \]

\[ B^*(S) = \sum_{V \in S} B(V) \quad (3.6) \]

This message is calculated by marginalization, and the operation is called projection.

Next, clique \( U \) absorbs from clique \( V \) as follows,

\[ B^*(U) = \frac{B(U) \times B^*(S)}{B(S)} \quad (3.7) \]

The operation represented by the above formula is called absorption.

In a more extended junction tree, the procedures \texttt{DistributeEvidence} and \texttt{CollectEvidence} establish the propagation process. A detailed discussion of these methods can be found in [35, 38]. We proceed with an outline of the methods mentioned. The methods are explained in an object-oriented style, where the objects are the cliques in the junction tree.

Let each node in the junction tree possess a separate \texttt{DistributeEvidence} and \texttt{CollectEvidence} processes which might be called by one of its neighbors.

**DistributeEvidence**

After a change in the belief table of clique \( U \), all neighbors calibrate to \( U \) which will force the neighbors’ neighbors to calibrate etcetera until the boundaries or leaves of the junction tree have been reached. This process of \texttt{DistributeEvidence} can be described in an object-oriented style as follows. Upon receiving a \texttt{DistributeEvidence} call, a node calibrates to the caller and calls \texttt{DistributeEvidence} to its own neighbors (with the exception of his caller).

**CollectEvidence**

After the belief tables have been changed, the belief table of an arbitrary clique \( V \) is made consistent with the changes by calling the method \texttt{CollectEvidence}. Similar to the process of \texttt{DistributeEvidence}, the process of collecting evidence consists of a series of calibrations. Again following the object-oriented style, upon receiving a \texttt{CollectEvidence} call, a node calls the method \texttt{CollectEvidence} in its neighbors. The recursively constructed series of \texttt{CollectEvidence} calls end at the leave nodes which have no neighbors except the caller. In this case, the caller calibrates to this boundary node and is finished with collecting evidence.
Global propagation

In order to restore consistency in the whole junction tree after some belief tables have been changed, it is sufficient to select an arbitrary node and call CollectEvidence followed by a call of DistributeEvidence. Finally, the belief tables are normalized and the marginal probabilities of the variables contained in the cliques are calculated.

Theorem 3.5.1 (Global propagation) Let \( T(B) = (C, \Phi) \), with \( C = (C\ell(G_u), E(G)) \) be a junction tree as defined in Definition 3.4.10. Let \( C\ell q_i \in C\ell(G_u) \), and let \( V_i \rightarrow C\ell q_i \). Now suppose the probability distribution \( P(V_i) \) has changed to \( P^{*}(V_i) \) as a result of the observation of evidence.

The consistency among \( C\ell(G_u) \) is restored after a collecting phase effectuated by calling CollectEvidence(\( C\ell q_j \)) followed by distributing phase effectuated by calling DistributeEvidence(\( C\ell q_j \)), where \( C\ell q_j \in C\ell(G_u) \).

The proof of the theorem is found in [35].

3.6 Propagation in junction trees - example

We consider the Bayesian belief network as shown in Figure 3.4.a and the junction tree as shown in Figure 3.4.b.

The propagation is conducted by the methods CollectEvidence and DistributeEvidence. These methods are called in the nodes of the junction tree. In order to indicate which method is called an argument is added, for example, CollectEvidence(\( C\ell q_i \)) is a call to the method CollectEvidence in \( C\ell q_i \).

3.6.1 Initialization

Prior to the propagation of the impact of evidence, the junction tree must be initialized. Initialization of the junction tree amounts to the propagation of prior probabilities. The propagation is established by calling CollectEvidence(\( C\ell q_i \)), where \( C\ell q_i \) is an arbitrary clique. After \( C\ell q_i \) has finished collecting evidence, DistributeEvidence(\( C\ell q_i \)) is called.

In Figure 3.5, the series of CollectEvidence calls that emerge from the initial call CollectEvidence(\( \{A, B, C\} \)) is shown by the solid curved arrows. The direction of the arrow indicates the flow of calibration, for example an arrow from \( C\ell q_i \) to...
$C_{lq_{i}}$ indicates that $C_{lq_{j}}$ calibrates to $C_{lq_{i}}$. In addition, the arrows are labeled with the following number of the calibration. Similarly, the labeled dashed curved arrows indicate the series of DistributeEvidence calls that emerge from the initial call DistributeEvidence($\{A, B, C\}$).

1. Clique $\{A, B, C\}$ calls CollectEvidence($\{G, C\}$)

   - Clique $\{G, C\}$ is a leaf and calculates the marginal probability distribution of its intersection with $\{A, B, C\}$ via projection as follows.

   $$B^1(C) = \sum_{\{G\}} \varphi_{\{G, C\}}$$

   (a) Clique $\{A, B, C\}$ calls CollectEvidence($\{C, D\}$)

   - Clique $\{C, D\}$ calls CollectEvidence($\{D, E, F\}$)

     * Clique $\{D, E, F\}$ is a leaf and calculates the marginal probability distribution of its intersection with $\{C, D\}$ via projection as follows.

     $$B(D) = \sum_{\{E, F\}} \varphi_{\{D, E, F\}}$$

     After this projection, clique $\{C, D\}$ is finished with collecting evidence and restores consistency via absorption as follows.

     $$B(\{C, D\}) = \varphi_{\{C, D\}} \times \frac{B(D)}{\varphi(D)}$$

   - Clique $\{C, D\}$ calculates the marginal probability distribution of its intersection with $\{A, B, C\}$ via projection as follows.

     $$B^2(C) = \sum_{\{D\}} \varphi_{\{D, C\}}$$

     After this projection, clique $\{A, B, C\}$ is finished with collecting evidence and restores consistency via absorption as follows.

     $$B(\{A, B, C\}) = \varphi_{\{A, B, C\}} \times \frac{B^1(C)}{\varphi(C)} \times \frac{B^2(C)}{\varphi(C)}$$

1. DistributeEvidence($\{A, B, C\}$)
3.6 Propagation in junction trees - example

- Clique \( \{A, B, C\} \) calculates the marginal probability distribution of its intersection with \( \{G, C\} \) and \( \{C, D\} \) via projection as follows.

\[
B^*(C) = \sum_{\{A,B\}} B(\{A, B, C\})
\]

- Clique \( \{G, C\} \) and \( \{C, D\} \) restore consistency via absorption as follows.

\[
B^*(\{G, C\}) = \frac{B(\{G, C\}) \times B^*(C)}{B^1(C)}
\]

\[
B^*(\{C, D\}) = \frac{B(\{C, D\}) \times B^*(C)}{B^2(C)}
\]

In clique \( \{G, C\} \) a leave has been reached and distribute evidence stops.

2. Clique \( \{A, B, C\} \) calls \texttt{DistributeEvidence}(\{C, D\})

- Clique \( \{C, D\} \) calculates the marginal probability distribution of its intersection with \( \{D, E, F\} \) via projection as follows.

\[
B^*(D) = \sum_{\{C\}} B(\{C, D\})
\]

- Clique \( \{D, E, F\} \) restore consistency via absorption as follows.

\[
B^*(\{D, E, F\}) = \frac{B(\{D, E, F\}) \times B^*(D)}{B(D)}
\]

In clique \( \{D, E, F\} \) a leave has been reached and distribute evidence stops.

The clique potentials that belong to Figure 3.4 are calculated as follows.

\[
\varphi_{\{A,B,C\}} = \mathcal{P}_A(A) \times \mathcal{P}_B(B) \times \mathcal{P}_G(G \mid A \land B),
\]

\[
\varphi_{\{C\}} = \mathcal{I},
\]

\[
\varphi_{\{G,C\}} = \mathcal{P}_G(G \mid C),
\]

\[
\varphi_{\{C,D\}} = \mathcal{P}_D(D \mid C),
\]

\[
\varphi_{\{D\}} = \mathcal{I},
\]

\[
\varphi_{\{D,E,F\}} = \mathcal{P}_E(E \mid D \land F) \times \mathcal{P}_F(F)
\]
\( \mathcal{P} = \{ \mathcal{P}_A(A), \mathcal{P}_B(B), \mathcal{P}_C(C \mid A \land B), \mathcal{P}_G(G \mid C), \mathcal{P}_D(D \mid C), \mathcal{P}_E(E \mid D \land F), \mathcal{P}_F(F) \} \)

\( \Phi = \{ \varphi_{A,B,C}, \varphi_C, \varphi_{G,C}, \varphi_{C,D}, \varphi_D, \varphi_{D,E,F} \} \)

\textbf{a:} Bayesian belief network, \( B = (G, \mathcal{P}) \). \hspace{1cm} \textbf{b:} Junction tree, \( T = (C, \Phi) \).

Figure 3.4: The Bayesian belief network and the junction tree.

\( \mathcal{P} \)
\( \mathcal{P}_B(B) \)
\( \mathcal{P}_C(C \mid A \land B) \)
\( \mathcal{P}_G(G \mid C) \)
\( \mathcal{P}_D(D \mid C) \)
\( \mathcal{P}_E(E \mid D \land F) \)
\( \mathcal{P}_F(F) \)
\( \Phi \)
\( \varphi_{A,B,C} \)
\( \varphi_C \)
\( \varphi_{G,C} \)
\( \varphi_{C,D} \)
\( \varphi_D \)
\( \varphi_{D,E,F} \)

Figure 3.5: The flow of calibrations during both \texttt{CollectEvidence} (solid arrows) and \texttt{DistributeEvidence} (dashed arrows).
3.6 Propagation in junction trees - example

Numbers

We give a numerical example of the calibration by writing out the second and third operation involved in the collect evidence face as discussed above. That is we apply the following formulas.

$$B(D) = \sum_{(E,F)} \varphi_{(D,E,F)}$$  \hspace{1cm} (3.8)

and

$$B(\{C,D\}) = \varphi_{\{C,D\}} \times \frac{B(D)}{\varphi(D)}$$  \hspace{1cm} (3.9)

We assumed all variables to be binary, for example $Sp(A) = (a, \neg a)$. The following probability assessment functions are used in the example.

$$P(E \mid D \land F) = \begin{cases} \left( P(e \mid d \land f), P(\neg e \mid d \land f) \right), \\ \left( P(e \mid d \land \neg f), P(\neg e \mid d \land \neg f) \right), \\ \left( P(e \mid \neg d \land f), P(\neg e \mid \neg d \land f) \right), \\ \left( P(e \mid \neg d \land \neg f), P(\neg e \mid \neg d \land \neg f) \right) \end{cases}$$

$$= \left( \begin{array}{c} 0.8, 0.2 \\ 0.7, 0.3 \\ 0.7, 0.3 \\ 0.1, 0.9 \end{array} \right)$$

$$P(F) = (P(f), P(\neg f))$$

$$= (0.9, 0.1)$$

$$P(D \mid C) = \begin{cases} \left( P(d \mid c \land f), P(\neg d \mid c) \right), \\ \left( P(d \mid \neg c), P(\neg d \mid \neg c) \right) \end{cases}$$

$$= \left( \begin{array}{c} 0.6, 0.4 \\ 0.1, 0.9 \end{array} \right)$$

The potential tables for the cliques $\{D, E, F\}$ and $\{C, D\}$ are shown in Table 3.6.1.

Applying Formula 3.8 yields the marginal belief table.

$$B(D) = (B(d), B(\neg d))$$

$$= (0.72 + 0.07 + 1.8 + 0.03, 0.63 + 0.01 + 0.27 + 0.09)$$

$$= (2.62, 1)$$
<table>
<thead>
<tr>
<th>$C^*_{{D,E,F}}$</th>
<th>$\varphi_{{D,E,F}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d\ e\ f$</td>
<td>$0.8 \times 0.9 = 0.72$</td>
</tr>
<tr>
<td>$d\ e\ \neg f$</td>
<td>$0.7 \times 0.1 = 0.07$</td>
</tr>
<tr>
<td>$d\ \neg e\ f$</td>
<td>$0.2 \times 0.9 = 1.8$</td>
</tr>
<tr>
<td>$d\ \neg e\ \neg f$</td>
<td>$0.3 \times 0.1 = 0.03$</td>
</tr>
<tr>
<td>$\neg d\ e\ f$</td>
<td>$0.7 \times 0.9 = 0.63$</td>
</tr>
<tr>
<td>$\neg d\ e\ \neg f$</td>
<td>$0.1 \times 0.1 = 0.01$</td>
</tr>
<tr>
<td>$\neg d\ \neg e\ f$</td>
<td>$0.3 \times 0.9 = 0.27$</td>
</tr>
<tr>
<td>$\neg d\ \neg e\ \neg f$</td>
<td>$0.9 \times 0.1 = 0.09$</td>
</tr>
</tbody>
</table>

Table 3.1: The potential tables for the cliques $\{D, E, F\}$ and $\{C, D\}$.

<table>
<thead>
<tr>
<th>$C^*_{{C,D}}$</th>
<th>$B({C, D})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d\ c$</td>
<td>$0.6 \times \frac{262}{1} = 1.57$</td>
</tr>
<tr>
<td>$d\ \neg c$</td>
<td>$0.1 \times \frac{1}{1} = 0.1$</td>
</tr>
<tr>
<td>$\neg d\ c$</td>
<td>$0.4 \times \frac{262}{1} = 1.04$</td>
</tr>
<tr>
<td>$\neg d\ \neg c$</td>
<td>$0.9 \times \frac{1}{1} = 0.9$</td>
</tr>
</tbody>
</table>

Table 3.2: Belief table of clique $\{C, D\}$ after the first calibration during the collection phase.

Applying Formula 3.8 yields the joint belief table as shown in Table 3.6.1.

The belief table $B(\{C, D\})$ is not a probability table, because the beliefs do not sum to 1; $1.57 + 0.1 + 1.04 + 0.9 = 3.61$. After the complete global propagation has been finished all belief tables are normalized in order to make them probability tables. The procedure to normalize a clique, $Clq_i$, is denoted by $\text{Normalize}(Clq_i)$.

### 3.6.2 Propagation of evidence

**Evidence**

Without loss of generality, we restrict our discussion to findings. In the junction tree, a finding $V_i = v_i$ is entered in the belief table $B(Clq_i)$, where $V_i$ is assigned to clique $Clq_i$, by giving all entries that correspond to an impossible configuration in $C^*_{Clq_i}$ based on the finding that $V_i = v_i$ a probability 0. We note that, in the
resulting table, the entries do not sum to 1. The sum of the remaining probabilities in the probability of the finding entered, \( P(V_i = v_i) \). By dividing the entries in the resulting table by \( P(V_i = v_i) \) the new probability table is established.

**Propagation**

Suppose a finding \( F = f \) has been observed. Entering this finding in clique \( \{D, E, F\} \) yields the following belief table.

<table>
<thead>
<tr>
<th>( C^*_{{D, E, F}} )</th>
<th>( B^*{{D, E, F} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d, e, f )</td>
<td>( B{{d, d, f} ) = 0.72</td>
</tr>
<tr>
<td>( d, e, \neg f )</td>
<td>0</td>
</tr>
<tr>
<td>( d, \neg e, f )</td>
<td>( B{{d, d, f} ) = 1.8</td>
</tr>
<tr>
<td>( d, \neg e, \neg f )</td>
<td>0</td>
</tr>
<tr>
<td>( \neg d, e, f )</td>
<td>( B{{d, d, f} ) = 0.63</td>
</tr>
<tr>
<td>( \neg d, e, \neg f )</td>
<td>0</td>
</tr>
<tr>
<td>( \neg d, \neg e, f )</td>
<td>( B{{d, d, f} ) = 1.27</td>
</tr>
<tr>
<td>( \neg d, \neg e, \neg f )</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.3: The belief table of clique \( \{D, E, F\} \) after finding \( F = f \) has been entered.

After entering the finding, a global propagation restores consistency in all cliques. Note that now the belief tables are used in the calculations instead of the potential tables.

### 3.7 Discussion

In the first part of this chapter, the method of probabilistic inference in singly connected Bayesian belief networks has been outlined. In addition, the intercausal dependency relations have been analyzed. The results of the analysis show that, in order to guarantee typical intercausal dependency relations such as explain away or joint explain, the assessment functions involved must obey constraining conditions. We note that in the modeling process, the constraining conditions mentioned could be used to restrict the assessment functions when certain intercausal dependency relations are wished.
In the second part of this chapter, the method of probabilistic inference in (undirected) junction trees has been discussed in some detail. The junction tree technique enables the propagation of probabilities in Bayesian belief networks without restrictions on the topology of the directed acyclic graph. We note that other methods are available and mention the method of conditioning [47] and the method of probabilistic inference in directed clique trees [43]. The latter method is similar to the junction tree method only the symmetry of the propagation of probabilities between two cliques is less clear. In addition, the propagation follows the structure of the tree regardless what evidence has been entered, which is a disadvantage in cases where we would like to directed the propagation process to updated one or some cliques only. The former method aims at transforming the multiple connected Bayesian belief network into a singly connected one by cutting through the loops. To this end a suitable cutset of variables must be found. We notice that finding an (optimal) cutset is equally difficult as finding the optimal triangulation [60, 59].
Chapter 4

Problem-directed decomposition

4.1 Introduction

This chapter addresses the problem of controlling the computational complexity of probabilistic inference in Bayesian belief networks. We recall from the introduction that probabilistic inference in arbitrary Bayesian belief networks is NP-hard [14], which means that it is very unlikely that efficient general-purpose algorithms will be found. As applications grow larger, the computational time required for an update of the associated network becomes rather time consuming. This is a result of both the size of the network, that is the number of variables in the network, and the topology of the associated graph. Concerning the latter, we note that the computational complexity of propagation and updating increases with the complexity of the graphs.

We note further that very large applications exist that permit modeling as simple graphs, such as MUNIN [3]. In other very large applications, the structure of the model is rather artificial in the sense that it is not only motivated by the application domain but also by considerations about the computational complexity of the final model, see for example [31, 58]. Because the modeling process in itself is already tremendously complex, it is undesirable that during the course of the modeling process one has to bear in mind issues irrelevant to the modeling task such as the computational complexity of the operative model. Therefore, we emphasize that optimization with respect to the computational complexity must be addressed independently and separated from the modeling process. In addition, the effects of relaxing the modeling task could well be that future models more naturally reflect the domain knowledge.
The *problem-directed decomposition* method proposed in the present chapter aims to utilize knowledge about a specified *problem instance* in order to save computations irrelevant to the *solution* of the problem instance. To this end, the network is transformed and probabilistic inference is restricted to a smaller network composed of a subset of independent components. The transformed version, called the *relevant network*, is constructed by exploiting the dependence and relevance relations among the variables in the original graph with respect to the specified problem instance. Problem-directed decomposition detects variables without impact on the query variables directly from the topology of the network, merely by examining the chains by which evidence and query vertices are connected.

In addition, new independence relations that emerge as a result of observations are explicitly represented in the graph. For this purpose we adopted the method of evidence absorption as introduced by Shachter [55] and later incorporated in Pearl's propagation algorithm by van der Gaag [66]. Very large savings of computations may be expected if the order of computational complexity is reduced according to the topology of the transformed network in comparison with the topology of the original network; the detection of components irrelevant to the solution of the problem instance postpone the need for computationally expensive triangulations. Moreover, the transformed network matches better with human comprehension capacities as a result of the focusing.

This chapter is organized as follows. The design considerations underlying the decomposition procedure are discussed in Section 4.2. In Section 4.3 a problem instance is defined and discussed in the light of the design considerations. The incorporation of evidence absorption in problem-directed decomposition is the subject of Section 4.4, and in Section 4.5 the final decomposition into relevant network and rest network is discussed. In Section 4.6, the handling of problem instance increments are discussed. Finally, in Section 4.7, the chapter is concluded with a discussion of the proposed problem-directed decomposition method.

### 4.2 Design considerations

The following requirements guided the development of the method of problem-directed decomposition.

- Exactness
- Generality
4.4 Problem instance

- Flexibility

From the previous section it is clear that we assume that problem-directed decomposition is applied after the domain model is fully specified. We choose to insist on exact probabilistic inference in order to avoid losing available domain knowledge. We add to this that, as demonstrated in [15], approximate probabilistic inference does not solve the computational complexity problem.

**Generality** means that there is no restriction to the Bayesian belief network topology nor to the problem instances that can be specified on the associated graph, in order to apply the decomposition method. As a result, in general, the components of the decomposed relevant network possess arbitrary topologies, that is, they may contain multiply connections (loops).

**Flexibility** means that the problem instance specification can dynamically be changed. The decomposition already obtained can be adapted to cope with problem instance increments without the need to perform the decomposition procedure all over again. A problem instance update may consists of incremental evidence, incremental queries and query decrements.

4.3 Problem instance

It is clear that the specification of a problem instance is essential for problem-directed decomposition. In this section we formally define a problem instance.

A problem instance specified on the graph associated to a Bayesian belief network $B$ consists of two disjunct sets of variables. The variables of interest, the query variables $Q \subset V(G)$, and the observed variables, the evidence variables $E \subset V(G)$.

**Definition 4.3.1 (problem instance)** Let $B = (G, P)$ be a Bayesian belief network as defined in Definition 3.2.2.

A problem instance specified on $G$ is a tuple $(E, Q)$, where $E$ and $Q$ are two disjunct subsets of $V(G)$. $E$ is called the evidence set and $Q$ is called the query set.

The *solution* of a problem instance consists of the (updated) probability distributions of the query variables given the evidence.
4.4 Evidence absorption

The first step in the decomposition method is the absorption of the evidence contained in \( E \) to obtain the absorbed network. To this end, the method of evidence absorption is incorporated into the decomposition method.

The method of evidence absorption was introduced by Shachter [51, 55] as part of the algorithm to perform probabilistic inference in a Bayesian belief network, and it was later incorporated in Pearl's propagation algorithm by van der Gaag [66]. The characteristic property of the propagation algorithm is that the instantiated vertex is eliminated from the network, and during the course of probabilistic inference, the network is transformed to reflect both newly created dependence relations and newly created independence relations.

The creation of new independence relations is taken into account by the method of evidence absorption. Whereas, the creation of new dependence relations is taken into account by the method of evidence propagation. Evidence propagation basically consists of the arc modification operation called arc reversal. This operation accounts for the insertion of new arcs into the network and the calculation of accompanying conditional probabilities.

The decomposition method does not apply all Shachter's propagation algorithms, because, as noted by Pearl [47] and van der Gaag [66], the repeated application of arc reversal during the evidence propagation has several drawbacks. We mention that it is computationally expensive to eliminate a vertex from the network. And that, in addition, the propagation algorithm requires a global supervisor. As opposed to evidence propagation, evidence absorption can be performed in linear time, and all computations are local to the evidence vertex and its successors. We emphasize that by applying evidence absorption, only new independence relations are created. This means that the complexity of the graph only decreases and so does the computational complexity of probabilistic inference. As proved in [66], applying evidence absorption on a Bayesian belief network results in a (absorbed) Bayesian belief network. In other words, the notion of a Bayesian belief network is invariant under evidence absorption. Therefore, evidence absorption can be applied independently and before building a junction tree. We proceed with the introduction of our notational conventions.
4.4 Evidence absorption

4.4.1 Definitions and notations

Absorbing the evidence contained in the set \( E \), on a Bayesian belief network, \( B = (G, \mathcal{P}) \), affects both the directed acyclic graph \( G \) and the probability assessment functions \( \mathcal{P} \). Informally speaking, the absorbed graph \( G^E \) is obtained from \( G \) by eliminating for each \( E_i \in E \) the outgoing arcs. In addition, the probability assessment functions of the (former) successors of \( E_i \in E \) are adjusted to reflect the evidence. We first define the transformation of \( G \) into the absorbed graph \( G^E \).

**Definition 4.4.1 (Absorbed graph)** Let \( G = (V(G), A(G)) \) be a directed acyclic graph associated to the Bayesian belief network \( B \) as defined in Definition 3.2.2, and let \( (E, Q) \) be a problem instance specified on \( G \).

The absorbed graph \( G^E = (V(G^E), A(G^E)) \) with respect to \( (E, Q) \) is defined by,

- \( A(G^E) = A(G) \setminus \{ \cup_{E_i \in E} (E_i, V_j) \in A(G) \mid V_j \in \sigma_G(E_i) \} \)
- \( V(G^E) = V(G) \)

The vertices contained in the set \( \{ \cup_{E_i \in E} \sigma_G(E_i) \} \) are called virtual evidence vertices, and denoted by \( V^E \).

We now proceed with the definition of an absorbed Bayesian belief network, which contains the transformation of \( \mathcal{P} \) into \( \mathcal{P}^E \).

**Definition 4.4.2 (Absorbed Bayesian belief network)** Let \( B = (G, \mathcal{P}) \) be a Bayesian belief network with directed acyclic graph \( G = (V(G), A(G)) \) and conditional probability assessment functions \( \mathcal{P} \), and let \( (E, Q) \) be a problem instance specified on \( G \). Let \( e_{i,j} \) be the observed value for evidence vertex \( E_i \in E \).

The absorbed Bayesian belief network \( B^E = (G^E, \mathcal{P}^E) \) is defined by,

- \( G^E \) is the absorbed graph as defined in Definition 4.4.1
- \( \mathcal{P}^E = \{ \mathcal{P}(V_i \mid C_{\rho E}(v_i)) \mid V_i \in V(G) \} \) is the set of non-negative functions such that
  1. \( \mathcal{P}^E(V_j \mid C_{\rho E}(v_j)) = \mathcal{P}(V_j \mid C_{\rho E}(v_j) \setminus e_{i} \wedge E_i = e_{i,j}) \) for all vertices \( V_j \in \cup_{E_i \in E} \sigma_G(E_i) \)
  2. \( \mathcal{P}^E(V_k \mid C_{\rho E}(v_k)) = \mathcal{P}(V_k \mid C_{\rho E}(v_k)) \) for all vertices \( V_k \in V(G) \setminus \cup_{E_i \in E} \sigma_G(E_i) \)
For Definition 3.2.2 is follows directly that $B^E$ is a Bayesian belief network, see also [66].

Figure 4.1 depicts the Bayesian belief network and problem instance that are used to illustrate the various transformation steps of problem-directed decomposition. Figure 4.2 shows the absorbed Bayesian belief network of our running example. We assume all variables to be binary, that means a variable $V_i$ can either be $TRUE$ or $FALSE$ denoted by $v_i$ for $V_i = TRUE$ and $\neg v_i$ for $V_i = FALSE$.

$$\mathcal{P} = \{\mathcal{P}(x_1), \mathcal{P}(q_1), \mathcal{P}(q_2), \mathcal{P}(x_2 \mid x_1), \mathcal{P}(x_2 \mid \neg x_1), \mathcal{P}(x_3 \mid x_1 \land q_1), \mathcal{P}(x_3 \mid x_1 \land \neg q_1), \mathcal{P}(x_3 \mid \neg x_1 \land q_1), \mathcal{P}(x_4 \mid e_1 \land q_1 \land q_2), \mathcal{P}(x_4 \mid e_1 \land \neg q_1 \land q_2), \mathcal{P}(x_4 \mid \neg e_1 \land q_1 \land \neg q_2), \mathcal{P}(x_4 \mid \neg e_1 \land \neg q_1 \land q_2), \mathcal{P}(x_4 \mid \neg e_1 \land \neg q_1 \land \neg q_2), \mathcal{P}(x_5 \mid x_2 \land \neg x_3), \mathcal{P}(x_5 \mid \neg x_2 \land x_3), \mathcal{P}(x_5 \mid \neg x_2 \land \neg x_3), \mathcal{P}(x_6 \mid x_4), \mathcal{P}(x_6 \mid \neg x_4), \mathcal{P}(x_7 \mid e_1), \mathcal{P}(x_7 \mid \neg e_1), \mathcal{P}(x_8 \mid e_1 \land x_9), \mathcal{P}(x_8 \mid e_1 \land \neg x_9), \mathcal{P}(x_8 \mid \neg e_1 \land x_9), \mathcal{P}(x_8 \mid \neg e_1 \land \neg x_9), \mathcal{P}(x_9 \mid x_{12}), \mathcal{P}(x_9 \mid \neg x_{12}), \mathcal{P}(x_{10} \mid x_9), \mathcal{P}(x_{10} \mid \neg x_9), \mathcal{P}(x_{11} \mid x_{10} \land x_9), \mathcal{P}(x_{11} \mid x_{10} \land \neg x_9), \mathcal{P}(x_{11} \mid \neg x_{10} \land x_9), \mathcal{P}(x_{11} \mid \neg x_{10} \land \neg x_9), \mathcal{P}(x_{12}), \mathcal{P}(q_3 \mid x_8), \mathcal{P}(q_3 \mid \neg x_8)\}$$

Figure 4.1: The Bayesian belief network $B = (G, \mathcal{P})$ and problem instance ($\{Q_1, Q_2, Q_3\}, \{E_1\}$) that will serve as our running example.
$\mathcal{P}^E = \{\ldots, \mathcal{P}^E(x_4 \mid q_1 \land q_2 \land e_1), \mathcal{P}^E(x_4 \mid q_1 \land \neg q_2 \land e_1), \mathcal{P}^E(x_4 \mid \neg q_1 \land q_2 \land e_1), \mathcal{P}^E(x_4 \mid \neg q_1 \land \neg q_2 \land e_1), \mathcal{P}^E(x_7 \mid e_1), \mathcal{P}^E(x_8 \mid x_9 \land e_1), \mathcal{P}^E(x_8 \mid \neg x_9 \land e_1), \ldots\}$

Figure 4.2: The absorbed network $B^E = (G^E, \mathcal{P}^E)$, with $E = \{E_1 = e_1\}$ The dots in the specification of $\mathcal{P}^E$ indicate that the assessment functions that are not repeated are identical to those given in Figure 5.1.

### 4.5 Focusing

After the absorbed network, $B^E = (G^E, \mathcal{P}^E)$, has been constructed, the second step in the problem-directed decomposition process is that of focusing on that part of the (absorbed) network that could bear an impact on $Q$. To this end, a second transformation on $B^E$ is performed.

In this section, we first discuss the focusing transformation step that decomposes the absorbed graph $G^E$ into the relevant graph and the rest graph. The basic idea underlying the relevant graph construction presented here is the identification of all variables that may contribute to the propagation of the impact of $E$ on $Q$. Our algorithm identifies the relevant (sub)graph.

An algorithm to identify all independence relations implied by the topology of the graph has already been proposed by Geiger and Pearl [23]. They proposed an algorithm to identify all variables not d-separated from $Q$ given $E$. Shachter [54] showed that specific vertices, he called them barren nodes, can in addition be removed without altering the computation of the updated probabilities of $Q$. The implications for reducing the computational complexity of applying Geiger and
Pearl's algorithm followed by applying Shachter's algorithm, as a preprocessing step, prior to using standard algorithms for probability propagation, in order to solve a problem instance, has been shown by Baker and Boul
t [5]. These authors constructed a minimal computational equivalent subgraph consisting of query and evidence vertices plus all vertices that could affect the probability distribution of the query variables and the arcs connecting them. However, their method is not able to handle incremental evidence (evidence outside \( E \)) without resorting to the original graph.

A method that uses knowledge about the localization of related variables, and on this basis constructs sub-domains, has been proposed by Xiang, Poole and Beddoes [72]. Sub-Domains are based on the localization associated with specific tasks performed in the information acquiring process for medical diagnosis (history taking, physical examination and performing specialized tests). Further, a user (external knowledge) has to decide to shift attention from one sub-domain to another.

Problem-directed decomposition differs from the decomposition method proposed by Xiang, Poole and Beddoes in that the problem instance dictates the structure of the relevant network. The identification is performed automatically (without knowledge about localization). Further, no external guidance to control the focus of attention is required. The presented approach differs from Geiger and Pearl's algorithm in a sense that variables not d-separated from \( Q \), but without impact on the problem instance solution, are not included in the relevant graph. The difference between the presented approach and Shachter's method is that barren nodes need not receive special treatment in order to be removed because they are not included. Problem-directed decomposition differs from Baker and Boult's in that the handling of problem instance increments is supported.

4.5.1 Relevant graph and rest graph

This section is started with some definitions.

**Definition 4.5.1 (Converging and Diverging vertex)** Let \( G = (V(G), A(G)) \) be a directed acyclic graph, and let \( p = [V_0, V_1, V_2] \) be a chain in \( G \).

- **The vertex** \( V_1 \) **is called converging in** \( p \) **if and only if**: \( (V_0, V_1) \in A(G) \) and \( (V_2, V_1) \in A(G) \).

- **The vertex** \( V_1 \) **is called diverging in** \( p \) **if and only if**: 
\begin{itemize}
  \item \((V_i, V_0) \in A(G)\) and \((V_i, V_2) \in A(G)\), or
  \item \((V_i, V_0) \in A(G)\) and \((V_2, V_i) \in A(G)\), or
  \item \((V_0, V_1) \in A(G)\) and \((V_1, V_2) \in A(G)\).
\end{itemize}

**Definition 4.5.2 (Strongly independent and Weakly independent)** Let 
\(G = (V(G), A(G))\) be a directed acyclic graph with vertex set \(V = \{V_1, \ldots, V_n\}\) and arcs \(A(G)\). Let \(Z\) be a subset of vertices in \(G\), and let \(\{V_i, V_j\} \in V(G) \setminus Z\).

Vertex \(V_i\) and vertex \(V_j\) are strongly independent in \(G\) given \(Z\), \(I(V_i, Z, V_j)_G\), if and only if:

- Each chain in \(G\) between \(V_i\) and \(V_j\) contains a diverging vertex \(V_k \in Z\).

Vertex \(V_i\) and vertex \(V_j\) are weakly independent in \(G\) given \(Z\), \(I(V_i, Z, V_j)_G\), if (and only if\(^1\)):

- None of the converging vertices on the chains in \(G\) between \(V_i\) and \(V_j\), are contained in \(Z\), nor any descendant of such converging vertices is contained in \(Z\).

Informally speaking, the relevant graph is composed of all chains that can possibly be used by the inference process to conduct the impact of \(E\) on \(Q\). We recall that the requirement of flexibility allows evidence to arrive at any vertex of \(G^E\). Upon adding new evidence, a blocked chain \(p\) between a pair \(\langle V_i, V_j \rangle\), with \(V_i, V_j \in V(G)\), of weakly independent vertices can be activated. The activation of \(p\) results in an additional active chain between \(V_j\) and \(V_i\) in which case a new loop is introduced. The requirement of generality led to the choice of junction trees for probabilistic inference. During junction tree building, handling loops receives a lot of computational resources because the efficacy of loop handling determines the computational complexity of the probability propagation and updating.

With the introduction of new loops, after building a junction tree, we must resort to the original network and rebuild the junction tree. The problem-directed decomposition method obviates the computational complex step of rebuilding the junction tree, at the expense of including in the relevant graph chains between weakly independent pairs of evidence and query variables. Those chains are temporary blocked and can become active in the light of new evidence. In the process

\(^1\)Remember from Chapter 3 that there are probability assessment functions possible such that the only if part of the previous statement does not apply.
of identifying the relevant graph, therefore, the d-separation criterion is not applied, but, instead, ordinary graph separation dictates which vertex is included into the relevant graph. Therefore, determining the chain between evidence and query variables (evidence-query chains) can be done in the underlying undirected dependency graph. For this purpose, the absorbed graph is moralized. As a result, the relevant junction tree can easily be extended after incremental evidence has been observed as demonstrated in Section 4.6.1.

We proceed with the formal definition of the relevant graph, and the complementary graph, called the rest graph.

**Definition 4.5.3 (Relevant graph)** Let $G = (V(G), A(G))$ be a directed acyclic graph and let $(E, Q)$ be a problem instance specified on $G$.

The relevant graph $G_{(E,Q)}$ of $G$ with respect to $(E, Q)$ is the undirected graph induced by $V^{(E,Q)} \cup JV$ on the absorbed moral graph $G^E_m$.

- $V^{(E,Q)}$ consists of all vertices $V_i \in V(G^E_m)$ such that $V_i$ is an element of a simple chain in $G^E_m$ between a pair $(E_j, Q_i)$, with $Q_i \in Q$ and $E_j \in E$. Such a chain is called an evidence-query chain.
- $JV = \{V_i \mid \exists (V_i, V_k) \in E(G^E_m) \text{ such that } (V_i \in (V(G) \setminus V^{(E,Q)})) \land V_k \in V^{(E,Q)}\}$. $V_i$ is called a joining vertex.

Figure 4.3 depicts the relevant graph, $G_{(E,Q)}$, of the running example.

![Figure 4.3: The relevant graph $G_{(E,Q)}$.](image-url)
We note that the process of evidence absorption results in a decomposition of $G$ into $G^E = \{G_1, G_2, \ldots, G_m\}$, by the elimination of all outgoing arcs of the evidence vertices. In relation to a problem instance $(E, Q)$, the graph $G^E$ can be partitioned further into the subset of components that possess at least one (virtual) evidence vertex and at least one query vertex, and into the subset of the remaining components. The components in the latter subset are strongly independent of the relevant graph $G^{(E,Q)}$. The vertices contained in the strongly independent components are denoted by $I^*$. In $G^E$, the strongly independent components are disconnected from $G^{(E,Q)}$. None of the variables $I^*$ can ever bear an impact on the solution of the problem instance.

We now define the rest graph as follows.

**Definition 4.5.4 (rest graph)** Let $G = (V(G), A(G))$ be a directed acyclic graph and let $(E, Q)$ be a problem instance specified on $G$. Let $G^{(E,Q)}$ be the relevant graph of $G$ with respect to $(E, Q)$.

The rest graph $J^{(E,Q)}$ of $G$ with respect to $(E, Q)$ is the graph induced by $V(G) \setminus (V^{(E,Q)} \cup I^*) \cup JV$ on the absorbed moral graph $G^E_m$.

We note that the vertex sets $V(G^{(E,Q)})$ and $V(J^{(E,Q)})$ overlap, i.e. $V(G^{(E,Q)}) \cap V(J^{(E,Q)}) = JV$.

Figure 4.4 depicts the rest graph of our running example.

![Diagram](image)

Figure 4.4: The rest graph $J^{(E,Q)}$ composed of two connected components. The joining vertices are shown dashed.

In the sequel, we assume, without loss of generality, that the original graph $G$ is connected.

Next we prove a property on the connections between the components of the relevant graph and the components of the rest graph. We state that, in $G^E_m$,
the connection between a component $K \in G^{(E,Q)}$ and a component $H \in J^{(E,Q)}$ consists of precisely one joining vertex. To formalize this, we prove the following lemma.

**Lemma 4.5.1 (Joining vertex)** Let $G = (V(G), A(G))$ be a connected directed acyclic graph associated to the Bayesian belief network $B$ as defined in Definition 3.2.2, and let $(E, Q)$ be a problem instance specified on $G$ as defined in Definition 4.3.1. Let $G^{(E,Q)}$ be the relevant graph of $G$ with respect to $(E, Q)$ as defined in Definition 4.5.3, and let $J^{(E,Q)}$ be the rest graph of $G$ with respect to $(E, Q)$ as defined in Definition 4.5.4.

In the absorbed moral graph $G_m^{(E)}$, the connection between the components $K \in G^{(E,Q)}$ and $H \in J^{(E,Q)}$ consists of precisely one joining vertex.

**Proof by contradiction.**

It suffices to prove that the assumption that there is a connection between $K \in G^{(E,Q)}$ and $H \in J^{(E,Q)}$ consisting of two joining vertices leads to a contradiction. (A connection consisting of more than two joining vertices includes a connection composed of two joining vertices.)

Assume that the connection between $K$ and $H$ consists of two joining vertices, $V_i, V_j$, with $V_i \neq V_j$. We prove that this assumption leads to a contradiction by analyzing the links that connect the joining vertices with the remaining vertices of $K$. Let these links be $(V_i, V_j), (V_i, V_j) \in A(K)$. We distinguish the following two situations.

1. $V_k = V_i$ (Figure 4.5.a)
2. $V_k \neq V_i$ (Figure 4.5.b)

We first prove that situation 2, $V_k \neq V_i \land V_i = V_j$, leads to a contradiction. By definition the component $H \in J^{(E,Q)}$ is connected and, therefore, there is at least one simple chain between vertex $V_j$ and vertex $V_i$.

Further, by definition, both $V_k$ and $V_i$ are on at least one evidence-query chain between a pair $(E_i, Q_j)$, with $E_i \in E$ and $Q_j \in Q$. But then, all elements on the simple chains between $V_i$ and $V_j$ in $H$ are on the chains between elements in $E$ and elements in $Q$, and, therefore, included in $G^{(E,Q)}$.

We prove that situation 1, $V_k = V_i$, leads to a contradiction. Assume that there are no links that connect the joining vertices to the remaining vertices of
4.5 Focusing

$K$ other than $(V_i, V_k)$ and $(V_j, V_k)$. Then, the evidence-query chain that contains $V_i$ is not simple since it contains two copies of vertex $V_k$. But, then, $V_i$ is not included in $G^{(E,Q)}$. A similar contradiction is found if we consider vertex $V_j$.

Next, we prove that the addition of links, $(V_i, V_i)$ or $(V_j, V_i)$, $V_i \in V(K)$ does not resolve the contradiction. Suppose we add a link $(V_j, V_i) \in E(K)$, $V_i \neq V_k$ (Figure 4.6). Now, there exists a simple chain $[V_i, V_j, V_k]$, and $[V_i, V_i, V_k]$ is contained in an evidence-query chains since both $V_k$ and $V_i$ are contained in $G^{(E,Q)}$. Then, $V_i$ is also on an evidence-query chain, since, again by definition, there is at least one simple chain between vertex $V_j$ and vertex $V_i$. Following this chain, vertex $V_k$ can be reached via $(V_i, V_k)$. But, then, the first situation is constructed, which means that $V_i$ and $V_j$ are included in $G^{(E,Q)}$. □

The next lemma states that the relevant graph as defined in Definition 4.5.3 provides a dependency model that enables the propagation of the impact the variables contained in $E$ have on the variables contained in $Q$.

![Diagram](attachment:image.png)

Figure 4.5: The two different connections composed of two joining vertices (gray) between $H \in J^{(E,Q)}$ and $K \in G^{(E,Q)}$, and $\{V_k, V_i, V_j, V_j\} \subset V(K)$ and $\{V_i, V_j\} \subset V(H)$.

![Diagram](attachment:image.png)

Figure 4.6: The addition of a link to $E(K)$.

**Lemma 4.5.2 (Relevant dependency structure)** Let $B = (G, P)$ be a Bayesian belief network as defined in Definition 3.2.2. Let $(E, Q)$ be a problem
instance specified on $G$ as defined in Definition 4.3.1. Let $G^E_m$ be the moralized version of the absorbed graph, $G^E$, as defined in Definition 4.4.1. Let $G^{(E,Q)}$ be the relevant graph of $G$ with respect to $(E,Q)$, and let $J^{(E,Q)}$ be the rest with respect to $(E,Q)$ as defined in Definition 4.5.4.

The dependency model defined by $G^{(E,Q)}$ is sufficient to enable the propagation in $B$ of the impact of the variables contained in $E$ on the variables contained in $Q$.

Proof.

To prove the lemma, it suffices to consider a problem instance consisting of one evidence variable and one query variable.

We consider $E_i \in E$ and $Q_j \in Q$. A necessary condition to enable the propagation of the impact $E_i$ has on $Q_j$ is that, in $G^E_m$, a chain $p$ between $E_i$ and $Q_j$ exists, because a partition of $G^E_m$ into, say, $G_1$ and $G_2$ such that $V(G_1) \cap V(G_2) = \emptyset$ results in the independence relation, $I(E_i, \emptyset, Q_j)_{G^E_m}$. We note that the presence of $p$ is not a sufficient condition for $E_i$ to have an impact on $Q_j$ since the independence relation $I(E_i, \emptyset, Q_j)_{G^E_m}$ may hold.

To prove the lemma, we demonstrate that in $B$ the propagation of the impact $E_i$ has on the variables contained in $V(J^{(E,Q)})$ does not result in an impact of the variables contained in $V(J^{(E,Q)}) \setminus V(J^{(E,Q)})$. We consider $G_1 \in G^{(E,Q)}$ with $E_i, Q_j \in V(G_1)$ and $G_2 \in J^{(E,Q)}$. From Lemma 4.5.1, it follows that the intersection of $V(G_1)$ and $V(G_2)$ contains one vertex, say $V_i$. We distinguish between the two cases; $G_2$ is singly connected and $G_2$ is multiply connected.

- $G_2$ is singly connected

From the propagation formulas for singly connected networks (Equation 3.4, Equation 3.5, Equation 3.3 and Equation 3.2) it follows directly that a message sent over an arc in one direction does not evoke a message to be sent over the same arc in the opposite direction. Thus the property stated in the lemma holds.

- $G_2$ is multiply connected

Figure 4.7 shows a portion of $G_2$ that contains a loop $l = [V_k, V_m, \ldots, V_n, V_k]$. We distinguish between two types of loop.

- In $G$ and as part of $l$, $V_k$ is a diverging vertex.

In this case, in any partition of $l$ into the chains $l_1 = [V_k, V_m, \ldots, V_o]$ and $l_2 = [V_k, V_n, \ldots, V_r]$ such that $V_i \in Adj_{G_2}(V_o)$, only one of them is active. Suppose that $l_1$ is active, then $I(V_k, \emptyset, V_o)_{G_2}$ due to the converging vertex $V_c \in l_2$. As a consequence, the loop can be cut by the elimination of one of the arcs $(V_o, V_c)$ or $(V_b, V_c)$ with $V_b, V_c \in Adj_{G_2}(V_c)$, without altering the impact of $E_i$ on $V(G_2)$. This makes $G_2$ singly connected, thus the property stated in the lemma holds.

- In $l$, $V_k$ is a converging vertex.

In this case, it may be impossible to partition the loop $l$ into an active and a blocked chain as described above in order to cut the loop to obtain a singly connected graph. However, to enable the impact $E_i$ has on $V_k$ to have an impact on $V_k$ after the propagation through $l$, the cliques in the junction tree representation $T(G_2)$ that contain subsets of the variables contained in $l$ must be connected with links such that each link includes $V_k$ in its label. Since clique $\{V_b, V_m, V_n, \ldots\}$ communicates with clique $\{V_m, V_m, \ldots\}$ only via a link labeled with $(V_n, V_m)$, the above condition cannot be fulfilled, and the property stated in the lemma holds.

\[\square\]

![Diagram](image.png)

Figure 4.7: Portion of the multiply connected directed acyclic graph $G_2 \in J^{(E,Q)}$. The dashed line between two vertices indicate the presence of a chain that connect them.

### 4.5.2 Algorithm

In this section, we discuss the identification of the decomposition of the moral absorbed graph $G^E$ into relevant graph and rest graph. In the process, special
vertices in $V(G^E)$ receive marks, and further information is added to $G^E$ in order to recover traced chains. In the discussion of the algorithm, we assume that the information mentioned can be represented in $G^E$. This means that we augment the graphs to include, for each vertex, a Boolean variable $eq(V_i)$ that indicates the containment of $V_i$ in the relevant graph, a pointer variable $pc(V_i)$ that points to a predecessor of $V_i$, and a Boolean variable $visit(V_i)$ that marks a visited vertex. The initial values are as follows.

$$\forall V_i \in V(G) : visit(V_i) = FALSE$$
$$\forall V_i \in V(G) : pc(V_i) = NULL - pointer$$
$$\forall V_i \in E \cup Q : eq(V_i) = TRUE$$
$$\forall V_i \in V(G) \setminus (E \cup Q) : eq(V_i) = FALSE$$

We assume $G^E_m$ to be composed of several connected components denoted, $G^E_m = \{G_1, G_2, \ldots, G_n\}$. It is clear that each component can be processed separately. Let us consider component $G_i$. Further, from the definition of the relevant graph (Definition 4.5.3), it is clear that if $V(G_i)$ does not contain at least one (virtual) evidence vertex and at least one query vertex, no processing is needed. Therefore, let us assume that $E_j \in V(G_i)$ and $Q_l \in V(G_i)$. Further, let $G_i'$ be the graph induced on $G_i$ by the vertex set $\{V_i \mid V_i \in V(G_i) \land eq(V_i) = TRUE\}$. The decomposition is realized by the two recursive procedures $eqChains(Q_l, G_i)$ and BackTracing $(E_j, G_i)$.

$eqChains$

The procedure $eqChains(Q_l, G_i)$ performs a depth-first search [20] in $G \setminus G_i'$ in order to identify all simple chains between $Q_l$ and the remaining vertices. Starting in $Q_l$, the depth-first process traverses all edges from vertex to vertex, visits the vertices and halts.

In the process, information about the direction in which an edge is traversed is added to $G_i$. For this purpose, predecessor pointers are set to point from the next vertex to the currently visited vertex. Further, each visited vertex is marked with $visit(.) = TRUE$. The $pc(.)$ pointers, when followed in the direction they point, identify the current chain from some element of $G'$ to the current vertex.

If the next vertex to visit has already been visited, then a loop has been detected. Since we are interested in identifying simple chains only, the edge
(current vertex, next vertex) is eliminated from \( G_i \). The edge elimination assures that only simple chains are identified. We note that edges eliminated from \( G_i \) may be included in the relevant graph because any edge induced by \( V^{(E,Q)} \) is included in the relevant graph.

In the case where \( V(G_i) \) contains multiple query vertices, one is selected to initiate the depth-first search. We note that the other query vertices are already marked with \( eq(.) = TRUE \) and, therefore, all chains between the (virtual) evidence vertices included in \( V(G_i) \) and \( Q_i \) have been found as well as all chains between \( Q_i \) and the other query vertices.

**BackTracing**

The procedure \( \text{BackTracing}(E_j, G_i) \) identifies the evidence-query chains by tracing the predecessor pointers starting at vertex \( E_j \). The vertices encountered receive marks \( eq(x) \) to indicate their containment in the relevant graph. The procedure is called, if during the depth-first search a vertex contained in the boundary between \( G_i \) and \( G_i \backslash G_i' \) is encountered. The boundary is the vertex set \( \{ V_i \mid V_i \in V(G_i') \land (\exists V_k \in \text{Adj}(V_i) \mid V_k \in V(G_i)) \} \). The identified simple chain connects two vertices of the relevant graph \( G^{(E,Q)} \).

Since one call to procedure \( \text{eqChains}(Q_i, G_i) \) suffices, only one depth-first search through \( G_i \) is performed. As a consequence, each link in \( G_i \backslash G_i' \) is traversed once and each link in \( G_i' \) is traversed twice. These observations result in a computational complexity for the determination of the relevant graph and the rest graph which is proportional to the number of edges in the original graph.

**Example**

Consider Figure 4.8 that depicts the component induced by \( \{ X_1, Q_1, Q_2, X_3, X_3, X_4^{E_1}, X_5, E_1, X_6 \} \) on the moral absorbed graph of our running example. In the sequel, we call this component \( G_1 \).

Suppose the procedure \( \text{eqChain} \) is initiated at vertex \( Q_1 \), and proceeds as indicated in Figure 4.9—Figure 4.13. The visited vertices are gray, and they are marked \( visit(.) = TRUE \). The directions of specific edges indicating the predecessor pointers, \( pe(.) \) is added. Further, the numbers associated to the thus-formed arcs indicate the order in which the links are traversed.

Upon traversing edge 5, a loop is detected since \( X_2 \) has already been visited.
The link \((X_2, X_3)\) is eliminated from \(G_1\), and the depth-first search continues as shown in Figure 4.9.b.

Figure 4.9: Effects of the procedures eqChain and BackTracing.

Upon traversing edge 6, again a loop is detected. The link \((X_1, X_3)\) is eliminated from \(G_1\), and the depth-first search continues as shown in Figure 4.10.a.

Upon traversing edge 7, a evidence-query chain is detected and the procedure BackTracing is called in \(X_3\). In Figure 4.10.b, the vertices included in the traced chain are shown black. They are marked with \(eq(.) = TRUE\). (Vertex \(Q_1\) is also shown black since, initially, \(eq(Q_1) = TRUE\).)

After the backtracing has finished, the depth-first search continues at the next unprocessed edge \((8)\) as shown in Figure 4.11.a.

Upon traversing edge 8 a second evidence-query chain is detected. The resulting situation after backtracing is depicted in Figure 4.11.b.

After the backtracing has been finished, the depth-first search continues at the next unprocessed edge \((9)\) also shown in Figure 4.11.b.
4.5 Focusing

Figure 4.10: Effects of the procedures eqChain and BackTracing, continued.

Figure 4.11: Effects of the procedures eqChain and BackTracing, continued.

Upon traversing edge 9, the boundary is reached. The depth-first search continues at the next unprocessed edge (10) as shown in Figure 4.12.a.

Upon traversing edge 10, a third evidence-query chain is detected. The resulting situation after backtracing is depicted in Figure 4.12.b.

After the backtracing has been finished, the depth-first search continues at the next unprocessed edge (11) also shown in Figure 4.12.b

Upon traversing edge 11, a fourth evidence-query chain is detected. However, the backtracing procedure does not add new vertices to the relevant graph.

The depth-first search continues at the next unprocessed edge (12) as shown in Figure 4.13.a.

Upon traversing edge 12, a fifth evidence-query chain is detected. The resulting situation after backtracing is depicted in Figure 4.13.b.

Since edge (12) was the last unprocessed edge, procedure eqChain halts.
Figure 4.12: Effects of the procedures eqChain and BackTracing, continued.

Figure 4.13: Effects of the procedures eqChain and BackTracing, continued. The component induced on $G_1$ by the black vertices is part of the relevant graph.

4.5.3 Relevant network and rest network

Given the decomposition of $G$ into $G^{(E,Q)}$ and $J^{(E,Q)}$, in this section we construct the Bayesian belief networks that fit the components of $G^{(E,Q)}$ and $J^{(E,Q)}$. To this end, for each component, probability assessment functions of the variables contained in that component must be determined. These induced assessment functions are defined in the following definition.

Definition 4.5.5 (Induced assessment functions) Let $B = (G, \mathcal{P})$ be a Bayesian belief network with directed acyclic graph $G = (V(G), A(G))$ and conditional probability assessment functions $\mathcal{P}$. Let $H$ be the graph induced by $V(H) \subseteq V(G)$ (Definition 4.5.3 induced graph).
4.5 Focusing

The induced probability assessment functions $\mathcal{P}^H$ are defined as follows.

- $\mathcal{P}^H(V_i | \rho_G(V_i)) = \mathcal{P}(V_i | \rho_G(V_i)), \forall V_i : \rho_G(V_i) \subseteq V(H)$
- $\mathcal{P}^H(V_i | \rho_G(V_i)) = 1$ (unity assessment function), otherwise

Definition 4.5.6 (Induced network) Let $B = (G, \mathcal{P})$ be a Bayesian belief network with directed acyclic graph $G = (V(G), A(G))$ and conditional probability assessment functions $\mathcal{P}$. Let $H$ be the graph induced on $G$ by $V \subseteq V(G)$.

The induced network $B_H = (H, \mathcal{P}^H)$ is defined by,

- $H = G_V$
- $\mathcal{P}^H$ is the set of nonnegative function as defined in Definition 4.5.5.

Definition 4.5.7 (Relevant network and rest network) Let $B = (G, \mathcal{P})$ be a Bayesian belief network with directed acyclic graph $G = (V(G), A(G))$ and conditional probability assessment functions $\mathcal{P}$, and let $(E, Q)$ be an problem instance specified on $G$. Let $V^{(E, Q)}$ be as defined in Definition 4.5.3. Let $G_{V^{(E, Q)}}$ be the directed graph induced on $G$ by $V^{(E, Q)}$. Let $J^{(E, Q)}$ be the rest graph of $G$ with respect to $(E, Q)$. Let $G_{V^{(J^{(E, Q)}}}$ be the directed graph induced on $G$ by $V(J^{(E, Q)})$. Let $V^{(E, Q)} \cup V^{(J^{(E, Q)}} = V_i$ be the joining vertex.

The relevant network $B^{(E, Q)}$ is defined to be the induced network $B_{G_{V^{(E, Q)}}}$ as defined in Definition 4.5.6.

To defined the rest network we distinguish between two types of connections.

- $\rho_G(V_i) \neq \emptyset$
  The rest network $B^{(E, Q)}$ is defined to be the induced network $B_{G_{V^{(J^{(E, Q)}}))}}$ as defined in Definition 4.5.6.

- $\rho_G(V_i) = \emptyset$
  The rest network $B^{(E, Q)} = (G_{V^{(J^{(E, Q)}}), \mathcal{P}^{G_{V^{(J^{(E, Q)}})}}})$ is defined by

1. $\mathcal{P}_{G_{V^{(J^{(E, Q)}}}}(V_i | \rho_G(V_i)) = \mathcal{P}^G(V_i | \rho_G(V_i)), \forall V_i \in V(J^{(E, Q)} \setminus \{V_i\})$
2. $\mathcal{P}^{G_{V^{(J^{(E, Q)}}}}(V_i) = 1$ (unity assessment function)

Both the relevant network and the rest network are Bayesian belief networks. This follows directly from Definition 3.2.2, Definition 4.5.3.

We recall that the exactness requirement states that the solution of the problem instance calculated in the relevant network must be identical to the solution of the problem instance calculated in the original network. This implies that the
exactness requirement demands that the impact of prior probabilities specified in the rest network on the joining vertices must be accounted for. Thus the junction tree of the relevant network $T(B^{(E,Q)})$, called the relevant junction tree, does not yet provide the adequate representation to solve the problem instance.

### 4.5.4 Active subnet

In this section, we determine which subnets in the rest network $B^{(E,Q)}$ have an impact on the solution of the problem instance.

We consider a subnet $H = (G_1, P^{G_1})$ in the relevant network and a subnet $N = (G_2, P^{G_2})$ in the rest network, such that $V(G_1) \cap V(G_2) = V_i$. In cases where $V_i$ separates $G_2$ into components, say, $\{G_{2,1}, \ldots, G_{2,n}\}$ we consider the impact of the induced subnetworks $B_{G_{2,i}}$ separately. Let $B_{G_{2,i}}$ be the subnetwork currently considered. For the sake of notational simplicity, we denote the subnetwork $B_{G_{2,i}}$, with $H = (G_3, P^{G_3})$.

In the discussion that follows, we assume without loss of generality that $B^{(E,Q)}$ consists of one network. A subnet $H$ that has an impact on the solution of the problem instance is called active. We note that, for the moment, the impact is due to prior probabilities in $H$. In Section 4.6.1, we return to the question of activeness in the context of incremental evidence observed in a component $H$.

We derive the necessary and sufficient condition on the topology of $G_{Adj(V_i)}$ that renders subnet $H$ active. Three topologies of $G_{Adj(V_i)}$ are distinguished as shown in Figure 4.14.a–c. In the figures we have, $\{X_1, \ldots, X_n, V_i\} \subset V(G^2)$ and $\{Y_1, \ldots, Y_m, V_i\} \subset V(G_3)$.

![Figure 4.14: Different topologies for $G_{Adj(V_i)}$.](image)

1. $\exists (X_i, V_i) \in A(G_1)$ (Figure 4.14.a)
Suppose \( r \in H \) is a root vertex, then any chain between \( V_i \) and \( r \) contains at least one converging connection, which is blocked due to the absence of diagnostic evidence. Thus, \( H \) is not active.

2. \( \forall X_i \in \text{Adj}_{G_1}(V_i) : (V_i, X_i) \in A(G_1) \)
   - \( \exists Y_j \in \text{Adj}_{G_2}(V_i) : Y_j \in \rho_{G_2}(V_i) \) (Figure 4.14.b)
     There is at least one active chain between a root vertex in \( H \) and \( V_i \).
     Thus, \( H \) is active.
   - \( \forall Y_j \in \text{Adj}_{G_2}(V_i) : Y_j \in \sigma_{G_2}(V_i) \) (Figure 4.14.c)
     Analogous to 1: \( H \) is not active.

The necessary and sufficient condition on the topology of \( G_{\text{Adj}(V_i)} \) that renders subnet \( H \) active is summarized in the following definition.

**Definition 4.5.8 (active subnet)** Let \( B = (G, \mathcal{P}) \) be a Bayesian belief network. Let \( G \) be decomposed into \( G_1 \) and \( G_2 \), such that \( V(G_1) \cap V(G_2) = \{V_i\} \). Let \( B_{G_1} \) and \( B_{G_2} \) be the induced Bayesian beliefs network as defined in Definition 4.5.6.

The subnetwork \( B_{G_2} \) is active with respect to the subnet \( B_{G_1} \) if and only if

\[ \forall V_i \in \text{Adj}_{G_1}(V_i) : (V_i, V_i) \in A(G_1) \text{ and } \exists y \in \text{Adj}_{G_2}(V_i) : y \in \rho_{G_2}(V_i) \]

**4.5.5 Upgrading the relevant junction tree**

In this section, we discuss the adjustment of the relevant junction tree to obtain the upgraded relevant junction tree \( T^w(K) \) in which the solution of the problem instance can be determined. Again consider \( K = (G_1, \mathcal{P}^{G_1}) \) and \( H = (G_2, \mathcal{P}^{G_2}) \), such that \( V(G_1) \cap V(G_2) = V_i \), as discussed in the previous section. And let \( H \) be an active subnet. Further, let \( T(K) = (\text{Cl}(G_{1,u}), \Delta) \) be the relevant junction tree and let \( T(H) = (\text{Cl}(G_{3,u}), \Gamma) \) be the rest junction tree.

Informally speaking, the adjustment amounts to multiplying some clique potentials in \( \Delta \) by the probability distribution of the associated joining vertex as calculated in the junction tree of the appropriate active component in the rest network. The probability distribution of the joining vertex captures the impact, prior probabilities have an impact on the solution of the problem instance.

Suppose that in \( T(H) \) the joining vertex \( V_i \) is included in clique \( C_l q_j \) and, in \( T(K) \), the joining vertex \( V_i \) is included in clique \( C_l q_i \), see Figure 4.15. In order to upgrade \( T(K) \) we perform the following procedure.
Figure 4.15: The communication channel between junction tree $T(H)$ and junction tree $T(K)$.

Algorithm 4.5.1 (Upgrading) Let $H = (G_3, \mathcal{P}^{G_3})$ be the rest network and let $K = (G_1, \mathcal{P}^{G_1})$ be the relevant network.

1. In $T(H)$, calculate the marginal probability of $V_i$. This probability is denoted by $P_H(V_i)$, and calculated in $T(H)$ as follows.
   (a) CollectEvidence ($Cl_{q_j}$)
   (b) Normalize ($B(Cl_{q_j})$)
   (c) Marginalize: $P_H(V_i) = \sum_{V_j \in Cl_{q_j}, V_i \neq V_j} B(Cl_{q_j})$

2. Transport $P_H(V_i)$ to $T(K)$ to obtain $T^n(K)$, by changing the clique potential of $Cl_{q_i}$ as follows.

\[ \varphi(Cl_{q_i}) = \prod_{V_i \in Cl_{q_i}, V_i \neq V_i} \mathcal{P}(V_i \mid \rho_{G_i}(V_i)) \times P_H(V_i) \]

After the clique potential had been changed according to Algorithm 4.5.1, we obtained $T^n(K)$. We note the omission of the distribution of the impact the variables contained in $V(G_1)$ have on the variables contained in $V(G_3)$. This omission does not influence the probability distribution $P_H(V_i)$ as stated in Lemma 4.5.2.

To formalize the procedure described above, we prove the following lemma.

Lemma 4.5.3 (Upgrading) Let $B = (G, \mathcal{P})$ be a Bayesian belief network, and let $T(B) = (Cl(G, \Phi))$ be a junction tree representation of $B$, and let $T(B)$ be initialized. Let $G$ be decomposed into $G_1$ and $G_2$ with $V(G_1) \cap V(G_2) = \{V_i\}$, such that the induced Bayesian network $B_{G_2}$ is an active subnet with respect to the induced Bayesian network $B_{G_1}$, as defined in Definition 4.5.8. Let $T(B_{G_1})$
and $T(B_{G_2})$ be junction tree representations associated to $B_{G_1}$ and $B_{G_2}$. Let $T^u(B_{G_1})$ be the upgraded junction tree as obtained after applying Algorithm 4.5.1. Finally, let $\Delta$ denote the set of potential functions in $T(B_{G_1})$, and $\Gamma$ denote set of the potential functions in $T(B_{G_2})$. Thus $T(B_{G_1}) = (Cl(G_{1,u}), \Delta)$ and $T(B_{G_2}) = (Cl(G_{2,u}), \Gamma)$.

In the initialized junction tree $T^u(B_{G_1})$, the belief tables of the cliques $Cl(G_{1,u})$ are identical to the belief tables of the corresponding cliques in the (initialized) junction tree $T(G)$.

Proof. The clique sets $Cl(G_{1,u})$ and $Cl(G_u)$ overlap because from Definition 4.5.3 it follows that $G_{1,u}$ is a subgraph of $G_u$. Likewise the clique sets $Cl(G_{2,u})$ and $Cl(G_u)$ overlap. The overlaps are such that $Cl(G_u) = Cl(G_{1,u}) \cup Cl(G_{2,u})$. Because of the equality in vertex sets of the cliques contained in $Cl(G_{1,u})$ and the corresponding cliques in $Cl(G_u)$ and the cliques contained in $Cl(G_{2,u})$ and the corresponding cliques in $Cl(G_u)$, we will denote corresponding cliques by identical identification. For example, we have $Clq_1 \in Cl(G_u)$ and $Clq_1 \in Cl(G_{2,u})$ contain the same set of vertices.

To prove the lemma, we demonstrate that, for corresponding cliques in $Cl(G_{1,u}) \cap Cl(G_u)$, the operations involved in the initialization of $T^u(B_{G_1})$ and the operations involved in the initialization of $T(G)$ are equal. Suppose that in $T(G)$ the connection between the clique sets $Cl(G_{1,u})$ and $Cl(G_{2,u})$ consists of the link labeled with $V_i$, and that this link connects the cliques $Clq_i \in Cl(G_{1,u})$ and $Clq_j \in Cl(G_{2,u})$.

Since the cliques in $Cl(G_1)$ communicate with the cliques in $Cl(G_2)$ only via the probability distribution of the variables $V_i$, we only need to consider the operations performed on the link labeled ($V_i$).

- We start with considering the operation performed in the initialization of $T(G)$.

The initialization of the junction tree $T(G)$ starts by calling CollectEvidence in an arbitrary clique, say $Clq_k \in Cl(G_u)$. We choose $Clq_k \in Cl(G_u) \setminus (Cl(G_{1,u}) \cap Cl(G_{2,u}))$. The call CollectEvidence($Clq_k$) results in a call CollectEvidence($Clq_i$) followed by a call CollectEvidence($Clq_j$), etc. When $Clq_j$ is finished with collecting evidence, $Clq_j$ calculates the probability distribution of the separation set $V_i$ by projection as follows.

$$P_B(V_i) = \sum_{V_i \in Clq_j \setminus \{V_i\}} B(Clq_i) \quad (4.1)$$
As a consequence, $\text{Clq}_i$ is finished with collecting evidence, and absorbs from $\text{Clq}_j$ as follows.

$$B^{*}(\text{Clq}_i) = \prod_{V_i \rightarrow \text{Clq}_i} \mathcal{P}(V_i | \rho_{G}(V_i)) \times P^*_B(V_i)$$

After which collecting evidence proceeds in the remaining cliques contained in $\text{Cl}(G_{1,u})$.

- Next we consider the initialization of the upgraded junction tree $T^u(B_{G_1})$.

Since $B_{G_2}$ is an active subnet with respect to $B_{G_1}$, and by Definition 4.5.5 we get the following equations.

$$\forall \text{Clq}_i \in \text{Cl}(G_1) : \delta_{\text{Clq}_i} = \prod_{V_i \rightarrow \text{Clq}_i \land V_i \neq V_i} \mathcal{P}(V_i | \rho_{G_2}(V_i)) \times \bar{I}$$

and

$$\forall \text{Clq}_k \in \text{Cl}(G_2) : \gamma_{\text{Clq}_k} = \varphi_{\text{Clq}_k}$$

In the upgrading process of Algorithm 4.5.1 step 1, the probability $P_{B_{G_2}}(V_i)$ is calculated after a collect evidence call in $\text{Clq}_j$. From the identity of the clique potentials, it follows directly that $P_{B_{G_2}}(V_i)$ is equal to $P^*_B(V_i)$ as calculated in Equation 4.1 in the process of collection evidence in $T(G)$.

In the upgrading process of Algorithm 4.5.1, step 2 is the transportation of $P_{B_{G_2}}(V_i)$ to $T(B_{G_1})$ as follows.

$$B(\text{Clq}_i) = \delta_{\text{Clq}_i} \times P_{B_{G_2}}(V_i) \tag{4.2}$$

From $P_{B_{G_2}}(V_i) = P^*_B(V_i)$, it follows that, in order to demonstrate that the right-hand part of Equation 4.1 equals the right-hand part of Equation 4.2, it suffices to prove that $\delta_{\text{Clq}_i} = \varphi_{\text{Clq}_i}$.

In $T(B_{G_1})$ vertex $V_i$ is assigned to $\text{Clq}_i$. From Definition 4.5.8 it follows that, in $G_1$, $V_i$ is a root vertex. And from Definition 4.5.5, it follows that, in $G_1$, the clique potential of $\text{Clq}_i$ contains the unity assessment function $\bar{I}$ to account for $V_i$. In $T(B_{G_2})$ and in $T(G)$, $V_i$ is assigned to $\text{Clq}_j$. From the above observations it follows that $\delta_{\text{Clq}_i} = \varphi_{\text{Clq}_i}$.

The identities demonstrated above prove the lemma. $\square$
Given $T_u(K)$, the solution of the problem instance is calculated by a collecting phase followed by a distributing phase after with the junction tree is initialized (Theorem 3.5.1) as follows.

**Algorithm 4.5.2 (Updating)**

1. Initialize ($T_u(K)$)
2. EnterFindings ($E$)
3. CollectEvidence ($Clique_k$), where $Clique_k$ is an arbitrary clique contained in $T_u(K)$
4. Distribute Evidence ($Clique_k$)
5. Normalize ($T_u(K)$)
6. For all $Q_i \in Q$ calculate the marginal probability distribution by marginalizations in appropriate cliques.

**Problem-directed decomposition in $H$**

In the foregoing discussion of the upgrade of $T(K)$, the junction tree of the complete active subnetwork $H$ was considered and the junction tree $T(H)$ was built. Obviously, problem-directed decomposition could also be applied in $H$ with problem instance; $Q = \{V_i\}$ and $E = R$, where $R$ is the set of root-vertices contained in $V(G_2)$. We note that after the upgrade and initialization of $T_u(K)$, the rest junction tree based either on $B_{G\wedge(n_iV_i)}$ or on $H$ is partially inconsistent with the initialized junction tree $T_u(K)$. For the impact of the variables contained in $E$ has not been propagated to the variables contained in the rest graph. Still, it is worthwhile to store the relevant junction ($T(B_{G\wedge(n_iV_i)}))$ for it can be used in the solution of a problem instance increment.

We end this section with a schematic overview of the affects problem-directed decomposition has on the junction tree structures that are built in the process depicted in Figure 4.16. In the figure, there are the following Bayesian networks. The original network, $B = (G, P)$; the relevant network with respect to the problem instance ($E, Q$), $B^{(E, Q)}$; the rest network with respect to the problem instance ($E, Q$), $B^{(E, Q)}$; the relevant part of an active subnet with respect to $B$ and ($E, Q$), $H_i$. And $\{T(H_i))\}$ denotes the set junction trees associated to the relevant parts of the active subnets, with respect to $B$ and ($E, Q$).
Problem-directed decomposition

\[ B \quad (E, Q) \]
\[ \text{problem-directed decomposition} \]
\[ \overline{B}^{(E, Q)} \quad B^{(E, Q)} \quad I^s \]
\[ \text{build junction tree} \]
\[ T(B^{(E, Q)}) \]
\[ \text{upgrade} \]
\[ \{T(H)\} \]

Figure 4.16: Schematic overview of the decomposition obtained with the method of problem-directed decomposition.

4.6 Problem instance increment

We have already mentioned that flexibility with respect to the problem instance is an important property of the method of problem-directed decomposition.

The present section discusses the procedures that enable the handling of problem instance increments. We discuss the handling of incremental evidence (Section 4.6.1), incremental queries (Section 4.6.2) and query decrement (Section 4.6.3).

We consider the original Bayesian belief network \( B = (G, P) \) with problem instance \( (E, Q) \) defined on \( G \). problem-directed decomposition has been applied on \( B \) and suppose that \( B \) is decomposed into subnet \( K = (G_1, P^{G_1}) \) in the relevant network \( B^{(E, Q)} \) and a subnet \( H = (G_2, P^{G_2}) \) in the rest network \( \overline{B}^{(E, Q)} \), such that \( V(G_1) \cap V(G_2) = V_i \).

Suppose further that, in \( T(K) \), the joining vertex \( V_i \) is included in clique \( C_{lq_j} \) and, in \( T(H) \), the joining vertex \( V_i \) is included in clique \( C_{lq_k} \).

In the discussion that follows, we assume without loss of generality that \( B^{(E, Q)} \) consists of one network, \( K \), and \( \overline{B}^{(E, Q)} \) consists of one network, \( H \). Further, let \( N = (G_3, P^{G_2}) \) be the relevant part of \( H \) obtained after applying the method of problem-directed decomposition in \( H \) with problem instance; \( Q = \{V_i\} \) and \( E = R \), where \( R \) is the set of root vertices contained in \( V(G_2) \). Finally, let \( \Gamma' \) be
the set of strongly independent variables.

We discuss problem instance increments consisting of one piece of evidence or one query. As a consequence of a problem instance increment, the decomposition of the original Bayesian belief network in the relevant network and the rest network and strongly independent variables changes. Each subsequent problem instance increment is handled in the changed decomposition in a similar way.

### 4.6.1 Incremental evidence

Incremental evidence is information about a change in the probability distribution of variable $e' \in V(G) \setminus \{E \cup Q\}$. We denote this variable by $e'$. Based on the procedure involved in handling the impact incremental evidence has on the solution of the problem instance, we distinguish between three types of incremental evidence.

1. $e' \in I^i$

   None of the variable in $I^i$ can bear an impact on the solution of the problem instance (Section 4.5.1).

2. $e' \in V(G^{(E,Q)})$

   The solution of the problem instance is updated using Algorithm 4.5.2 where step 1 now is: **EnterFindings** ($e'$).

3. $e' \in V(J^{(E,Q)})$

   In general, the impact of $e'$ on the solution of the problem instance depends on the impact of $E$ on $V(J^{(E,Q)})$. Since in the case where $H$ is active, the junction tree $T(N)$ has already been built for a subnet $N = (G_3, P^{G_2})$ of $H$, we distinguish between when $H$ is active and when $H$ is not active.

   - $H$ is active
     - $e' \in V(G_3)$
       The belief tables of the cliques contained in the $Cl(G_3)\setminus Clq_3$ are inconsistent with the belief tables of the cliques contained in $Cl(G_2)$. In general, the impact $e'$ has on the variables in $V(G_1)$ depends on the impact of $E$ on $V(G_3)$. Therefore, consistency between the
cliques contained in $\text{Cl}(G_1)$ and the cliques contained in $\text{Cl}(G_3)$ must be restored before the impact of $e^i$ on $V(G_1)$, in particular on $Q$, can be calculated. To this end, clique $\text{Clq}_j \in \text{Cl}(G_3)$ calibrates to $\text{Clq}_j \in \text{Cl}(G_1)$, and $\text{DistributeEvidence}$ is called in $\text{Clq}_j$ of $T(H)$. After consistency has been restored, the belief tables on the cliques contained in $\text{Cl}(G_1) \cup \text{Cl}(G_3)$ are identical to the corresponding cliques in the junction tree $T(B)$ after the propagation of the impact of $E$. The proof is similar to the proof of Lemma 4.5.2. The incremental evidence $e^i$ is entered in $T(H)$ and its impact on $V(G_1)$ is calculated as follows. Firstly, $\text{CollectEvidence}$ is called in $\text{Clq}_j$ in the junction tree $T(H)$, resulting in the updated belief table $B^*(\text{Clq}_j)$. Secondly, $\text{Clq}_i$ is forced to take over the probability distribution of the vertices it has in common with $\text{Clq}_j$ by a calibration of $\text{Clq}_i$ to $\text{Clq}_j$. Thirdly, the impact of the changed belief in $\text{Clq}_i$ is propagated to the remaining cliques contained in $\text{Cl}(G_1)$ by calling $\text{DistributeEvidence}$ in $\text{Clq}_i$. We note that only half of the operations involved in a global propagation through $T^u(K)$ are required, since only one belief table, i.e., the belief table of $\text{Clq}_i$, has changed.

After the distributing phase has been finished, the cliques contained in $\text{Cl}(G_1)$ are normalized and the probability distributions of the query variables that constitute the solution of the problem instance $(E \cup e^i, Q)$ are calculated by marginalizations in the appropriate cliques of $T^u(K)$. The algorithm of the described procedure is given in Algorithm 4.6.1.

- $e^i \notin V(G_3)$

Let $M = (G_4, P^{G_4})$ be the rest network of $H$ after problem-directed decomposition in $H$ with problem instance; $Q = \{V_i\}$ and $E = R$, where $R$ is the set of root vertices contained in $V(G_2)$. Suppose that $V(G_3) \cap V(G_4) = V_m$, is the joining vertex between subnet $N$ and subnet $M$. Suppose further that, in $T(N)$, the joining vertex $V_m$ is included in clique $\text{Clq}_r$, and, in $T(M)$, the joining vertex $V_m$ is included in clique $\text{Clq}_q$.

From the identification of $N$, it follows that the subnetwork $M$ is not active with respect to $M$. Thus for the joining vertex $V_m$ the local topologies as shown in Figure 4.14.a or Figure 4.14.c apply.
4.6 Problem instance increment

In the case where $V_m$ is a root vertex in $G_2$, $V_m$ is contained both in $Clq$, and in $Clq_2$. From Definition 4.5.7 we get the following probability assessment functions.

\[ P^G_2(V_i | \rho G_2(V_i)) = P^G_2(V_i | \rho G_2(V_i)), \forall V_i \in V(G_4) \setminus \{V_m\} \]

\[ P^G_2(V_m) = \overline{1} \text{ (unity assessment function)} \]

The propagation of the impact of $et$ on $V(G_1)$, in particular on $Q$, is accounted for as follows. Firstly, the junction tree $T(H)$ is upgraded. The upgrading procedure is analogous to the upgrading procedure applied to obtain $T^w(K)$ (Algorithm 4.5.1). Only now, the impact on the joining vertex $V_m$ is due to incremental evidence $et$.

After $T^w(N)$ has been determined, the procedure described in Algorithm 4.6.1 is evoked to calculated the solution of the problem instance $(E \cup et, Q)$.

- $H$ is not active

  The relevant subnet of $H$ with respect to $et$ must be identified. This is done by applying problem-directed decomposition in $H$ with problem instance; $Q = \{V_i\}$ and $E = \{et\}$. The junction tree $T(H^{(V_i), (et)})$ is built and upgraded (Algorithm 4.5.1) to obtain $T^w(H^{(V_i), (et)})$. Finally, the solution of the problem instance $(E \cup et, Q)$ is calculated by applying Algorithm 4.6.1.

Algorithm 4.6.1 (Incremental evidence in active subnet)

1. Calculate $P_K(V_i)$ by projection,

   \[ P_K(V_i) = \sum_{V_i \in Clq, V_i \neq V_i} B(Clq_i) \]

2. Calculate $P^*(Clq_j)$ by absorption,

   \[ P^*(Clq_j) = \frac{P(Clq_j) \times P_K(V_i)}{P_N(V_i)} \]

3. Restore consistency in $T(N)$ by calling DistributeEvidence($Clq_j$).

4. EnterFindings($et$)

5. CollectEvidence($Clq_j$)

6. Normalize ($Clq_j$)
7. Calculate $P_N(V_i)$ by marginalization:

$$P_N(V_i) = \sum_{V_i \in C_lq_j, V_i \neq V_1} B(C_lq_j)$$

8. Calibrate $C_lq_i$ to $C_lq_j$:

$$P^*(C_lq_i) = \frac{P(C_lq_i) \times P_N(V_i)}{P_K(V_i)}$$

9. Distribute Evidence ($C_lq_i$)

10. Normalize ($T^*(K)$)

11. For all $Q_i \in Q$ calculate the marginal probability distribution $P(Q_i)$ by marginalizations in the appropriate cliques.

4.6.2 Incremental query

Incremental query is the extension of the set of variables of interest. We denote the incremental query variable by $q' \in V(G) \backslash \{E \cup Q\}$.

Based on the procedure involved in handling the impact incremental query has on the solution of the problem instance, we distinguish between three types of incremental query.

1. $q' \in I^p$

The variables contained in $E$ do not have an impact on the variables contained in $I^p$. The probability distribution of $P(q')$, therefore, is calculated in the initialized junction tree of the Bayesian belief network induced by $G_1$, by marginalization in the appropriate clique. The solution of the problem instance $(E, Q)$ is extended with $P(q')$.

2. $q' \in V(G^{(E, Q)})$

The probability distribution of $P(q')$ is directly calculated in the junction tree $T^*(K)$, by marginalization in the appropriate clique. The solution of the problem instance $(E, Q)$ is extended with $P(q')$.

3. $q' \in V(J^{(E, Q)})$

The relevant graph must be extended to include all simple chains between $q'$ and $V_i$. The focusing procedure with inputs $\{V_i\}$ and $\{q'\}$ is called for
this purpose and identifies $G_5$. The relevant network $A = (G_5, P^{C_5})$ is constructed, and the junction tree $T(A)$ is built. The junction tree $T(A)$ will be part of $T^u(B^{(E,Q \cup q')})$, the junction tree associated to the relevant network with respect to the incremented problem instance $(E, Q \cup q')$. To incorporate the impact of active subnets with respect to $A$, the junction tree $T(A)$ is upgraded using Algorithm 4.5.1 to obtain $T^u(A)$. And $T^u(A)$ is initialized.

The contribution of $E$ to the probability distribution $P(q')$ is accounted for in $T^u(A)$ by changing the the belief table of $Clq_j$ as follows.

$$B^*(Clq_j) = B(Clq_j) \times P_K(V_i)$$

The updated probability distribution of $q'$ is calculated by calling CollectEvidence in $Clq_k$, followed by a call DistributeEvidence in $Clq_k$.

Concerning the distribution phase, we state that the probability distribution of the joining vertex is identical to the probability distribution of the joining vertex as calculated in the upgrade of $T(B^{(E,Q)})$.

Finally, the junction trees $T^u(B^{(E,Q)})$ and $T^u(A)$ are joined to obtain the junction tree, $T^u(B^{(E,Q \cup q')})$, by establishing the link labeled with $V_i$ between $Clq_i$ and $Clq_j$.

### 4.6.3 Query decrement

Query decrement is the reduction of the set of variables of interest. We denote the decremented query variable by $q' \in Q$. We consider the junction tree $T^u(B^{(E,Q)})$ where the solution of the problem instance $(E, Q)$ has been calculated.

In order to account for the query decrement, the junction tree $T^u(B^{(E,Q)})$ is pruned. This pruning consists of the elimination from the tree of of all cliques $Clq_k$ contained in $Cl(G^{(E,Q)})$ such that $E_i \notin Clq_k$ with $E_i \in E$ and and $Q_j \notin Clq_k$ with $Q_j \in Q \setminus q'$ and

1. $q' \in Clq_k$ or

2. $Clq_k$ is a clique of the chain from a clique that contains $q'$ to a clique that contains either $E_i \in E$ or $Q_j \in Q \setminus q'$.

Let $V^{(E,Q \setminus q')}$ be the set of variables contained in the cliques of the junction tree $T^u(B^{(E,Q)})$ after pruning as described above. Further, let $G^{(E,Q \setminus q')}$ be the
graph induced by $V^{(E,Q\backslash q')}$ on $G$. The pruned junction tree is identical to the junction tree associated to the relevant Bayesian belief network with respect to the problem instance $(E, Q\backslash q')$.

$$T^u(B^{(E,Q\backslash q')}) = (CL(G^u_{(E,Q\backslash q')}, P^{G_{(E,Q\backslash q')}}))$$

### 4.7 Discussion

With the development of realistically sized Bayesian belief networks, it has become increasingly important to have methods that limit the propagation of probabilities to specific parts of the model only. The problem-directed decomposition method proposed here reduces the number of propagation-related computations involved in solving a problem instance without losing the flexibility to incorporate incremental evidence.

To this end, the original network is decomposed into a relevant network and a rest network. The decomposition is determined in time proportional to the number of edges in the moral graph. The junction tree associated to the relevant network calculates the solution of the problem instance such that the updated probability distributions of the query variables are identical to the updated probability distributions of these variables, as calculated in the junction tree associated to the complete domain model. In addition, the relevant junction tree forms a necessary base for the calculation of the problem instance in the sense that it enables problem instance increments without the necessity of resorting to the original graph and rebuilding the junction tree. The generality of the method enables its application in many circumstances. When making use of the proposed method, significant savings in computations may be expected if the graph is composed of densely connected components which are sparsely inter-connected, and the problem instance is contained in a limited number of components.

The research reported in the chapter has also been presented at two different conferences [62, 64].
Chapter 5

Experimental evaluation

5.1 Introduction

In the previous chapter we proposed the problem directed decomposition method. We recall that the main objective of problem directed decomposition is to save on the computational effort involved in probabilistic inference in large Bayesian belief networks. In order to decide as to whether or not the obtained increase in performance is worthwhile depends on the application, and an analysis in the discerned domain and application at hand is needed. However, an interesting question is: What increase in performance can be expected on the average for a particular class of network and problem instance settings? This question gives rise to a second one: On which criteria do we base our discrimination between different classes. In other words, how can the classes be defined? The present chapter addresses these questions and is organized as follows: The first question is addressed by an experimental evaluation of the average behavior of the method of problem directed decomposition. The problem stated in the second question is touched upon in Section 5.2, where we discuss the classes we consider in the reported experiments. In Section 5.3 we motivate the experimental approach taken, while the discussion of the performed experiments can be found in Section 5.4. The results of the experiments are reported in Section 5.5. And in Section 5.6, the experimental evaluation with different randomly generated networks is completed with conclusions and final remarks. Strongly motivated by the experimental results with different classes of randomly generated networks, we performed a set of experiments with the method of problem directed decomposition in two real-life networks. The discussion of the experiments with real-life networks is found in
Section 5.7 and concludes this chapter.

5.2 Classes

To gain insight into the impact of the problem directed decomposition method in the average case we must have at our disposal a test bed composed of a reasonable number of large real-life networks. We note that these networks are preferably not already modeled to meet the computational demands of probabilistic inference. Unfortunately, as mentioned in the previous chapter, this test bed is not yet available. For lack of the real-life test bed, we designed experiments with different classes of \textit{randomly} generated networks.

We define a class of randomly generated networks to be the collection of all networks with directed acyclic graphs that possess the same ratio between the number of vertices and the number of edges.

In addition, all variables are assigned identical state space sizes. We then assume, without loss of generality, that all variables are binary.

We are aware of the fact that networks containing binary variables only may deviate from real-life networks. However, the defined classes serve the purpose to enable the study of the impact of problem directed decomposition in the average case in relation with the graphical part of the network.

We refrained from randomly generating state spaces because our primary interest is to investigate the increase in performance as a result of applying problem directed decomposition to the combination of network \textit{topology} and problem instance setting.

5.3 Why, and what experiments?

From a theoretical point of view, the best case and the worst case can easily be identified. A worst case situation occurs when evidence is obtained only for vertices without outgoing arcs, and all other vertices represent query variables. In this case, applying problem directed decomposition is useless as it results in a relevant network which is identical to the original network. In the best case, evidence is obtained for all vertices adjacent to a query variable. In this case, applying problem directed decomposition results in a decomposition of the original network into a relevant network composed of components that contain query
and evidence vertices only, such that the states of the queries are completely
determined by the surrounding evidences.

For the above observation, we conclude that the best and worst case are asso-
ciated with very typical combinations of network and problem instance settings.
As a result, the bounds provided by the best and worst case are too broad be-
of any practical use, and they do not provide any "feel" of the behavior of prob-
lem directed decomposition in general. We, therefore, concentrate on the more
interesting average case.

We would like to emphasize that the motivation for the experimental evalua-
tion lies in the fact that, even for randomly generated networks, the increase in
performance cannot be predicted purely analytically and based on theoretical con-
siderations about properties of the network and problem instance settings. One
of the difficulties is that the increase in performance may be strongly dependent
on the global topological properties of the network, that is the number and the
extensiveness of loops. Another difficulty is that the increase in performance also
strongly depends on the distribution of the query and evidence variables over the
network in relation to the networks topology.

Simultaneously, we note that the observed behavior in randomly generated
networks can at most be considered as an indication for real-life situations. Ac-
cordingly, the experiments discussed here should be considered as a first explo-
ration of the behavior of the method of problem directed decomposition.

5.3.1 The expected behavior

The impact of the method of problem directed decomposition is quantified as the
ratio between the computational effort involved in probabilistic inference in the
junction tree associated with the original network and probabilistic inference in the
junction tree associated with the relevant network. This ratio is called the gain.

The gain is a result of the two aspects of problem directed decomposition:
evidence absorption and focusing. We elaborate on the expected effects of the
aspects mentioned on the average gain separately, and formulate hypotheses to
be tested and quantified in the experiments.

Evidence absorption

- Varying the number of arcs, with a fixed number of evidences and a fixed
  number of queries.
For an increasing connectivity density of the networks, the average gain increases as a result of the arc elimination operation, since the average number of outgoing arcs per vertex increases.

However, with increasing connectivity, the average gain decreases because it becomes less likely that evidence absorption results in the formation of components. Stated differently, with increasing connectivity it becomes more likely that components are connected with multiple chains which must all contain an evidence variable in order to separate the components.

- Varying the number of pieces of evidence, with a fixed connectivity and a fixed number of queries.

For an increasing number of pieces of evidence, the average gain increases because more independence relations are made explicit.

**Focusing**

- Varying the number of arcs, with a fixed number of pieces of evidence and a fixed number of queries.

For an increasing connectivity density of the networks, the average gain decreases because there are more evidence-query chains.

- Varying the number of queries, with a fixed connectivity and a fixed number of pieces of evidence.

Again, for an increasing number of queries, the average gain decreases because there are more evidence-query chains.

- Varying the number of pieces of evidence, with a fixed connectivity and a fixed number of queries.

And again, for an increasing number of pieces of evidence, the average gain decreases because there are more evidence-query chains.

The behavior of the average gain we measured has arisen as a result of the combination of the various aspects discussed above.

**Combined effects of evidence absorption and focusing**

- For an increasing number of queries, the average gain decreases until saturation has been reached. We use the term saturation heuristically to denote
the situation where (almost) all chains of the original network are included in the relevant network. After saturation has been reached, the average gain remains more or less constant.

- For an increasing number of pieces of evidence, the average gain is influenced in two opposite directions.

1. The average gain decreases as a result of the formation of more evidence-query chains. Similarly to the situation of an increasing number of query variables, the relevant network approaches saturation. We note that, after saturation has been reached, the relevant network, in general, does not include all chains present in the original network as a consequence of the eliminated arcs. As a result, the average gain that belongs to saturation reached by increasing the number of entered pieces of evidence, in general, differs from a saturation point reached by increasing the number of query variables.

2. The average gain increases as a result of evidence absorption. We note that this increase is not linear with the number of pieces of evidence, for two reasons,
   
   - Arc elimination might result in the separation of a component from the relevant network.
   - Arc elimination might cut through loops.

We pose the following hypotheses and questions concerning the combined effect of evidence absorption and focusing have on the average gain.

- Concerning the influence of the connectivity density of the networks and the different saturations, we pose the following hypotheses.

Firstly, we consider the situation where the number of entered pieces of evidence is fixed, and increase the number of query variables.

**Hypothesis 1:** Saturation (S1) is reached sooner with increasing connectivity densities.

Secondly, we consider the situation where the number of query variables is fixed, and increase the number of entered pieces of evidence.

**Hypothesis 2:** Saturation (S2) is reached sooner with increasing connectivity densities.
Hypothesis 3: The value of the average gain that belongs to S1 is smaller than the value of the average gain that belongs to S2.

- Concerning the influence of the connectivity density of the networks and the course of the average gain we pose the following hypotheses and questions.

We consider the situation where the number of query variables is fixed, and increase the number of entered pieces of evidence.

Hypothesis 4: After saturation has been reached, the average gain increases.

Question 1: Before saturation has been reached, does the average gain decrease, increase or remains constant?

We consider the situation where the number of entered pieces of evidence is fixed, and increase the number of query variables.

Hypothesis 5: After saturation has been reached, the average gain remains constant.

Hypothesis 6: Before saturation has been reached, the average gain decreases.

Next, we discuss the experiments that should provide us with the rejection or confirmation of the hypotheses, and the answers to the questions. Further, the experiments give an indication of the values of the average gain for the investigated parameter setting.

5.4 Experiments

5.4.1 Design

We designed our experiments for network classes that are defined by a connectivity parameter only. By doing so, we avoided the design of network classes containing networks that are bound to yield "good" experimental results. We remark that real-life networks, too, have differing topological properties. Further, we choose to distribute the evidence and query variables randomly over the network. Again, the motivation for this was to avoid the construction of parameter settings especially suitable for providing "good" experimental result. However, the following is a taxonomy of problem instances based on the distribution of the evidence and query variables commonly used. In diagnostic problem instances, the queries are
located in the upper part of the network, that is they do not possess many pre-
decessors, and the pieces of evidence are located in the lower part of the network. In *prognostic* problem instances the reverse is true. We note, however, that many reasonable problem instances are not represented in this taxonomy.

### 5.4.2 Calculation of the gain

The gain of the method of problem-directed decomposition is calculated as the ratio between the computational effort involved in probabilistic inference in a junction tree associated with the original network and probabilistic inference in a junction tree associated with the relevant network. The computational effort is calculated as the total of all operations performed in the calibrations carried out in one global propagation. The number of operations in one calibration sum of the number of multiplications and the number of additions performed. See Equations 3.6,3.7.

We recall from Section 3.4 that the problem of finding the optimal triangulation is NP-complete [74]. Thus the best we can do is to search for suboptimal solutions by using heuristic search algorithms. As a result, a multiply connected network may have several different junction trees stemming from applying triangulation algorithms based on different optimization heuristics. We note that, depending on the problem to be solved, optimality may be defined differently.

We do not address which heuristic is optimal for what application. Instead, we apply the minimum size heuristic [49] in all cases. Informally speaking, the minimum size heuristic follows a greedy strategy for adding fill-in links (Algorithm A.0.2 and Theorem A.0.3). Each time, a fill-in link is added such that the size of the newly formed clique is minimal. We refer to [41] for a discussion of a range of triangulation heuristics.

In the experiments, we do not take into account the computational expense of building the junction trees mentioned. We also do not take into account the computational effort involved in the initialization of the junction trees. Concerning these simplifications, we note that the gain calculated is a little pessimistic. The relevant network might contain fewer loops than the original network and therefore fewer triangulations are required, simplifying the triangulation process. The argument that building the junction trees need only be done once, and therefore, can be considered as a static computational effort applies partially when problem directed decomposition is concerned. The gain problem directed decomposition
acquires comes mainly from the dynamic computational effort involved in probabilistic inference. However, (eternally) postponed triangulations obviously add to the gain; but in the experiments we do not take these effects into account. Finally, we recall that performing problem directed decomposition is computationally inexpensive since its computational complexity is proportional to the number of edges in the directed acyclic graph.

5.4.3 Set-up

The experiments are performed in the search space spanned by three parameters: the ratio of the number of vertices and the number of arcs in the generated network, the number of obtained pieces of evidence and the number of queries variables.

Concerning the first-mentioned parameter, we note that in randomly generated networks, the trends in the gain we want to investigate do not depend on the actual size of the network. Therefore, for each class, it suffices to consider networks of equal size. In our experiments, we considered networks consisting of 20 vertices, where each vertex represents a binary variable. A class of networks, then, is defined by the number of edges, $m \geq 1$, in the directed acyclic graphs of the networks that are contained in the class.

In each network, $k$ pieces of evidence are entered, $1 \leq k \leq 5$, and $l$ query variables are pointed out, $1 \leq l \leq 5$, such that a variable cannot be both a query and an evidence variable. The integers $m$, $l$ and $k$ differ in the various experiments.

Network generator

The random network is completely defined after the connected directed acyclic graph has been generated, since the network to be generated randomly comprises of binary variables only. For this purpose, a graph generator has been written in analogy to the graph generator reported to be used in similar experiments in [60]. For a specified class, a maximally connected directed acyclic graph is created as follows. Firstly, an ordered list of $n$ vertices is created, $V_1, V_2, \ldots, V_n$. Secondly, for each vertex $V_i$, arcs are inserted to point to higher numbered vertices in the list. This assignment of arcs results in a maximally connected acyclic directed graph.

Thirdly, arcs are eliminated until $m$ arcs are left. To preserve randomness, the elimination of an arc must be done at random. This elimination may cause the directed acyclic graph to fall apart into components. We choose to avoid the
5.4 Experiments

generation of disconnected networks to ensure that the generated problem instance is contained in (only) one component. The underlying hypothesis used here is that the evidence and query variables associated with the same problem instance are normally not spread over different components. We, therefore, generated one such component by introducing a connectivity test in the generator. We further note that the connectivity bias is commonly used in this type of experiment, see for example [60, 66]. The connectivity test consists of checking before eliminating an arc as to whether the resulting network would remain connected. Since not all arcs are equally likely to be removed, not all members of a class are equally likely to be generated. However, supported by the results of our experiments, we feel that a broad range of networks is generated.

5.4.4 Performed experiments

In this section, we discuss the experiments we performed and repeat our objectives in designing these experiments.

We recall that the experiments are performed in the search space spanned by three parameters: the ratio of the number of vertices (20) and the number of arcs in the generated network, the number of entered pieces of evidence and the number of query variables.

To study the influence of the density of connectivity on the gain obtained after applying the method of problem directed decomposition, we performed all our experiments with 5 classes defined by:

1. number of vertices = 20 and $m = 19$, singly connected networks
2. number of vertices = 20 and $m = 22$, multiply connected networks containing 3 simple loops
3. number of vertices = 20 and $m = 25$, multiply connected networks containing 6 simple loops
4. number of vertices = 20 and $m = 30$, multiply connected networks containing 11 simple loops
5. number of vertices = 20 and $m = 35$, multiply connected networks containing 16 simple loops
We refrain from considering more densely connected networks because real-life networks tend to be sparsely connected, as has been argued in, for example, [47].

To study the influence of the number of queries and pieces of evidence, for each class, we vary the number of queries from 1 to 5, $1 \leq l \leq 5$ and likewise we varied the number of pieces of evidence, $1 \leq k \leq 5$.

We do not consider problem instances with more than 5 queries or more than 5 pieces of evidence because, in networks comprising 20 vertices, this would mean that more than 50% of the variables are contained in a problem instance, and we expect that the contribution of finding the evidence-query chains to the gain will only be minor. In other words, we expect the gain to be determined by evidence absorption only. For an experiment evaluation of the impact of the method of evidence absorption we refer to [66]. The reported results, however, cannot directly be interpreted in the context of the experiments discussed in this chapter. The parameters we consider are the actual number of operations performed during probabilistic inference in the original and in the transformed network. We note that both networks might be multiply connected and that the method of junction trees is utilized.

In the experiment in [66], the considered parameters bear a direct relation to the actual number of operations performed during probabilistic inference if and only if Pearl's method of probabilistic inference in singly connected networks is applicable.

To enable the analysis of the average behavior of the method of problem directed decomposition, the following sets of networks and problem instances were generated. For each class and for each conceivable setting of the parameters $l$ and $k$, 100 networks were generated together with the appropriate problem instances. Thus, for each class a total of 2500 networks with associated problem instances were generated. Below, we refer to the set of 100 networks with the same parameter setting for $m, l$ and $k$ as $C_{m,l,k}$. 
5.5 Results and discussion

5.5.1 The data from the experiments

An inspection of the gain values associated with the networks in $C_{m,l,k}$, $m \in [19, 22, 25, 30, 35]$, $1 \leq l \leq 5$ and $1 \leq k \leq 5$, has led to the subdivision of their total range into two ranges. $1 \leq \text{gain} \leq 10$ and $10 < \text{gain} \leq \text{maximum gain}$.

The subdivision obviates that for a given class $C_{m,l,k}$, the averages are determined largely by only a very small fraction of the members of a class $C_{m,l,k}$.

In the discussion we concentrate on the subset of the members from each class with associated gain values within the first range. As shown in Table 5.1, for each considered class, the majority of the members fall into the first range.

The average gain (m), average deviation of the average gain (d) and the percentage of the networks with gain values within the specified range (s), of all experiments are collected and brought together in Table B.1–B.5 as shown in Appendix B.

<table>
<thead>
<tr>
<th>$m =$</th>
<th>$1 \leq \text{gain} \leq 10$</th>
<th>maximal average for a particular setting of $l$ and $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>93%</td>
<td>171</td>
</tr>
<tr>
<td>22</td>
<td>97%</td>
<td>249</td>
</tr>
<tr>
<td>25</td>
<td>98%</td>
<td>404</td>
</tr>
<tr>
<td>30</td>
<td>99%</td>
<td>794</td>
</tr>
<tr>
<td>35</td>
<td>99%</td>
<td>5161</td>
</tr>
</tbody>
</table>

Table 5.1: Percentage of class members with gain values within the first range and maximal average for the five considered network classes.

In addition, we make the following remarks on the nature of the data contained in Table B.1–B.5.

- When connectivity increases, the maximum gain increases and the percentage of members contained in the first range decreases.

- With an increasing number of queries the percentage of members contained in the first range increases.
• In singly connected networks, average gain values equal to 0 are found in
the second range for a relatively large number of queries and a relatively
small number of pieces of evidence.

• When connectivity increases, average gain values equal to 0 are found in
both ranges, again for a relatively small number of pieces of evidence, but
now also for a decreasing number of queries.

The explanation that applies to all effects noted above is that when the number
of queries increases, saturation is reached sooner when the connectivity increases.
After saturation has been reached, the average gain is established as a result of
evidence absorption only. This also explains why, in those cases, we found average
gain values of > 0 for a relatively large number of pieces of evidence.

Summing up, we state that the results of a closer analysis of the data contained
in the first range will provide us with an idea of the common behavior of the
method of problem directed decomposition in the classes of randomly generated
networks. We add that the results are a little pessimistic, due to the exclusion
of very high gain values associated with the typical combination of network and
problem instance setting, where applying problem directed decomposition is very
fruitful.

5.5.2 Tendencies

We chose to represent the data gathered from the experiments in two sets
of graphs. Each set contains a graph for each considered class. In the first set,
the average gain is plotted as a function of the number of query variables. For
each class, this function is plotted five times: for one piece of evidence, for two
pieces of evidence up to five pieces of evidence. In the second set, the average
gain is plotted again but now as the function of the number of pieces of evidence.
For each class, this function is plotted five times, now for different values of the
number of query variables. In our discussion, we consider the results as shown in
Figure 5.1 and Figure 5.2. Both figures are based on the same data but provide
different views of the trends we discuss next.

Firstly, the observed behavior in each class is discussed separately. We start
each discussion with the effect of varying the number of queries (Figure 5.1). We
chose this starting points because the effects of varying the number of queries
allow a discussion that is more or less isolated from the effects of varying the
5.5 Results and discussion

![Diagram](image)

Figure 5.1: Average gain over 100 randomly generated networks, plotted as a function of the number of queries.

Number of entered pieces of evidence. We then continue with discussing the same results, but now from the perspective of varying the number of pieces of evidence (Figure 5.2). Secondly, the trends in the behavior when there are increasing connectivity densities are discussed.
Figure 5.2: Average gain over 100 randomly generated networks, plotted as a function of the number of evidences.

**The class defined by 20 vertices and 19 edges**

- We consider $|E| \in [1, 5]$, and increasing $|Q|$.

A decrease in the average gain is found, see Figure 5.1.a. This decrease is explained by the formation of more evidence-query chains. When the value of $|E|$ increases,
5.5 Results and discussion

the decrease diminishes because the increasing effect of evidence absorption on the average gain outweighs (some of) its decreasing effect. Saturation has not yet been reached, and Hypotheses 6 is confirmed.

- We consider $|Q| \in [1, 5]$, and increasing $|E|$.

A decrease in the average gain followed by an increase has been found, see Figure 5.2.a. For $|Q| = 1$, the change from decreasing to increasing occurs at $|E| = 4$. For $|Q| \in [2, 3, 4]$, this change also occurs at $|E| = 3$. For $|Q| = 5$, this change also occurs at $|E| = 4$. Summarizing the observed behavior, we found that the decrease in the average gain diminishes, when the value of $|E|$ increases.

Concerning Question 1, we note that before saturation has been reached, a decrease in average gain is found.

The class defined by 20 vertices and 22 edges

- We consider $|E| \in [1, 5]$, and increasing $|Q|$.

A decrease in the average gain is found, see Figure 5.1.b. We note that this decrease is smaller than that given in Figure 5.1.a. Again, saturation has not yet been reached, and Hypotheses 6 is confirmed.

- We consider $|Q| \in [1, 5]$, and increasing $|E|$.

A constant or increasing average gain is found, see Figure 5.2.b. As expected, the smallest average gain is found for one piece of evidence. Also as expected, the average gain increases when the number of pieces of evidence increases. We observe a non-linearity: for low values of $|E|$, the increase in average gain is less than for higher values of $|E|$. This trend is explained as follows. For low values of $|E|$ the diminishing effect on the average gain, due to the formation of evidence-query chains, is still present. With the addition of pieces of evidence, saturation is approached; the diminishing effect disappears and the enlargement of the average gain, as a result of arc elimination, causes the observed effect on its own.

From these result we conclude that saturation is only reached after we find a non-increasing average gain for increasing values of $|E|$. We note that the underlying assumption here is that the non-linear effect of evidence absorption is averaged out. However, from the experimental results, from the classes later discussed, we learn that this assumption is not valid for more densely connected
networks. For example, fluctuations in the (increase in the) average gain are found even for \(| Q | = 5\) in the classes defined by 20 vertices and 25, 30 and 35 edges. The explanation for this observation is easily found in the erratic nature of the effect of evidence absorption on the average gain. When the connectivity density of the networks increases, it becomes more likely that evidence absorption results in loops being cut through. However, the likelihood of component separation diminished when the connectivity densities of the networks increased. The experimental results, indeed, show this unpredictable behavior of the average gain with respect to the number of entered pieces of evidence, and even more so for more densely connected networks.

Concerning Question 1, we note that before saturation has been reached, a constant or increasing average gain is found.

**The class defined by 20 vertices and 25 edges**

- We consider \(| E | \in [1, 5]\), and increasing \(| Q |\).

The decrease found in the average gain is only moderate, see Figure 5.1.c. Compared to Figure 5.1.b, again the decrease has diminished. We note that the fluctuations in the average gain for different values of \(| Q |\) stem from the erratic nature of the effect of entering pieces of evidence. We further note that the fluctuation increases when the number of entered pieces of evidence increases and when the connectivity density of the networks increases. In order to minimize the blurring effect of entered evidence when we want to detect saturation situations as a result of the number of queries, we consider the situations where only one piece of evidence has been entered.

- We consider \(| Q | \in [1, 5]\), and increasing \(| E |\).

An increasing average gain is found, see Figure 5.2.c. In addition, we found that the increase grows for increasing values of \(| E |\). This trend becomes even more apparent in the following classes.

**The class defined by 20 vertices and 30 edges**

- We consider \(| E | \in [1, 5]\), and increasing \(| Q |\).

A new trend is being established, see Figure 5.1.d. The average gain is constant for \(| E | = 1\); saturation has been reached, and Hypotheses 5 is confirmed.
5.5 Results and discussion

- We consider $|Q| \in [1, 5]$, and increasing $|E|$.

An increasing average gain is found, see Figure 5.2.d. We make the following observations: The distance between the equi-evidence points has increased, the fluctuations have become bigger, and the value of the average gain has increased, and has increased more so for increasing values of $|E|$, see Figure 5.1.d. Concurrently, the non-linearly is weakened. These trends are explained as follows. Starting with the last trend mentioned, we note that, when there is increasing connectivity, the relevant network reaches saturation for a lesser number of queries. Apparently, in the considered class, the relevant network had already approached saturation for parameters settings of one query variable and one piece of evidence. For this observation we infer that the topology of the considered networks is such that almost all vertices are contained in overlapping loops.

Concerning the increase in distance between the equi-evidence points, we note that, close to saturation, the increase in average gain (as a result of arc elimination in evidence absorption) increases when connectivity densities increase. This increase is explained by the average increase in the number of outgoing arcs per vertex. Hypothesis 4 is confirmed.

The class defined by 20 vertices and 35 edges

- We consider $|E| \in [1, 5]$, and increasing $|Q|$.

Similarly to the previous discussed class, a saturation point has been reached, see Figure 5.1.e.

- We consider $|Q| \in [1, 5]$, and increasing $|E|$.

An increasing average gain is found, and saturation points have been reached see Figure 5.2.e. The trends are similar to the trends discussed for the case of 30 edges, only more extreme.

The influence of increasing connectivity densities

Reconsidering the above discussions we find that, when the connectivity densities of the networks increase, the following observation can be made.

- The decrease in the average gain, as a result of the formation of evidence-query chains, diminishes.
• The increase in the average gain, as a result of evidence absorption, grows. This observation confirms Hypothesis 2.

• Saturation is reached for a decreasing number of query variables. This observation confirms Hypothesis 1.

Concerning Hypothesis 3, we make the following remark. In all considered classes we found that the average gain for $x$ pieces of entered evidence and for $y$ query variables, is more than or equal to the average gain for $y$ pieces of entered evidence and for $x$ query variables, where $x > y$. This implied the confirmation of Hypothesis 3.

Concerning Question 1, we have found that no statement can be made about the trends of the average gain that applies to all considered classes. For singly connected networks, we found decrease of the average gain followed by an increase. For increasing connectivity density we found constantly diminishing decrease.

## 5.6 Conclusion and remarks

Summarizing the experimental results, it can be stated that the increasing effect on the average gain as a result of focusing quickly declines when the connectivity of the randomly generated networks increases. Further, the increasing effect on the average gain as a result of evidence absorption, although erratic in nature, persists in all considered classes and increases with the increasing connectivity densities of the networks according to what can be expected. In addition, for classes with 20 vertices and a number of arcs that is equal to or exceeds 30, for one piece of entered evidence and one query variable, saturation has been reached.

Based on these results, it can be concluded that the randomly generated networks we used in our experiments, in classes with 20 vertices and a number of arcs that is equal to or exceeds 30, in the average case, all vertices are included in one or more loops. We note that in real-life networks, the percentage of vertices included in one or more loops, generally, is much less.

Further, for a particular network and problem instance setting, the increase in performance may be remarkable, since in all classes considered in our experiments extremely high values for the average gain have been found, albeit for a small percentage of the considered cases.

Concerning the ability of randomly generated networks to provide an insight into the impact of applying the method of problem directed decomposition, we
arrive at the conclusion that this ability is only limited. However, before we can do any better, a thoroughly investigation of all possible real-life networks will be necessary in order to define a taxonomy into classes.

We emphasize that the problems identified here will be actual in all kinds of experimental evaluations of heuristics using randomly generated network, where the topology of the networks is of major importance to the results of the experiments. Further, much recent research in the field of Bayesian belief networks aims at improving the computational effort involved in probabilistic inference and/or junction tree construction by means of heuristic methods [59, 66, 41, 5, 72, 73]. These heuristic methods can only be evaluated by experimentation. We, therefore, feel that it is worthwhile to make an effort in order to construct a general available test bed based on the yet to be determined network taxonomy. We envision this test bed to be composed either of the real-life networks labeled to belong to some class, or to be composed of the definitions of the classes in the taxonomy. The use of such a test bed would enable comparison of the results of all experiments. Obviously, a comparison of the results of various methods would increases the value of the evaluation dramatically.

To conclude this chapter, we return to the remark made in the introduction of the present chapter. “The best way to answer the question as to whether or not the obtained increase in performance is worthwhile, is to perform an analysis in the discerned domain and application at hand”. We find that despite the insight obtained from the reported experiments, we still fully support this statement.

5.7 Two real networks

In this section, we illustrate the method of problem directed decomposition in two particular real-life networks: the Central NeuroMuscular Failure (CNMF) network\(^1\), in the domain of neonatology, and the Alarm network [6], in the domain of anesthesia. Three problem instances were specified for both the CNMF network and the Alarm network. The relevant associated networks are determined by using the method of problem directed decomposition. For all networks, the original ones and the relevant ones, junction trees were constructed making use of the minimum size heuristic in the triangulation algorithm.

\(^1\)The CNMF model was built by the author in cooperation with Steen Andreassen of Aalborg University Center, and Carinke Buiting who was at that time employed at Leiden University.
In the discussions, we consider the impact of evidence absorption and focusing separately. To this end, we applied the two procedures of problem directed decomposition in reverse order. Firstly, focusing was applied and secondly, on the resulting network, evidence absorption has been applied. The network obtained after applying focusing is referred to as focused network.

5.7.1 CNMF

We discuss the effect of the method of problem directed decomposition on the CNMF network. The directed acyclic graph of the domain model is shown in Figure 6.3. The model possesses the following statistics related to the graph size and topology. The directed acyclic graph contains 35 vertices and 36 edges. The two non-overlapping simple loops contain 6 and 3 variables, thus approximately 25% of the variables are contained in a loop. In Appendix C, a specification of the state spaces of the contained variables is provided.

On inspection of the the model, we find that a subdivision into four submodels seems natural.

1. Drug administration to the mother.
2. Immaturity of the baby.
3. Possible trauma for the baby.
4. Expected status of the newborn.

The submodels mentioned coincide with focal areas in the modeled domain knowledge. Further, each focal area is separated from the rest of the network after the elimination of only one arc. This means that the focal areas are organized in a tree structure. In the directed acyclic graph of Figure 6.3 we distinguish the subgraphs induced by (1) the predecessors of the vertex *drugs* and the successors of these predecessors; (2) the predecessors of the vertex *glyamNM* and the successors of these predecessors; (3) the predecessors of the vertex *TraumaNM* and the successors of these predecessors, and (4) the vertex subset \{CNMFFailure, anoxia, art. ventilation, central cyanosis\}.

Subgraph (1) is separated by the elimination of arc (DrugsNM, CNMFFailure), subgraph (2) is separated by the elimination of arc (glyamNM, CNMFFailure), and subgraph (3) is separated by the elimination of arc (TraumaNM, TraumaNM – glyamNM).
Considered problem instances

The problem instances we consider are prognoses made in order to predict the probability of a baby cyossed at birth because of a temporary (or a number of temporary) condition(s). In the network, the variable Central cyanosis provides an indication for the oxygen content of the baby's blood, and represents the query variable we are interested in. The probability of Central cyanosis is used to anticipate what equipment might be required after the baby is born, for example, a ventilator and an incubator.

The method of problem directed decomposition has been applied with problem instances: \((E_1, Q), (E_2, Q)\) and \((E_3, Q)\), where

\[
\begin{align*}
E_1 &= \{\text{peth. to mother, time lapse, val. to mother, time lapse}\} \\
E_2 &= \{\text{prolonged labor, immaturity}\} \\
E_3 &= \{\text{Trauma Cause, Vitamine K, Steroid}\} \\
Q &= \{\text{Central cyanosis}\}
\end{align*}
\]

We note that the the above specification of problem instances indeed respects the partition of the model into the four focal areas, because each evidence set is contained in a separate focal area.

5.7.2 Results and discussion

<table>
<thead>
<tr>
<th>Problem directed decomposition</th>
<th>Operations (gain)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td></td>
</tr>
<tr>
<td>(B)</td>
<td>3679</td>
</tr>
<tr>
<td>(B^{(E_1, Q)})</td>
<td>518 (7.10)</td>
</tr>
<tr>
<td>(B^{(E_2, Q)})</td>
<td>1440 (2.55)</td>
</tr>
<tr>
<td>(B^{(E_3, Q)})</td>
<td>1120 (3.28)</td>
</tr>
</tbody>
</table>

Table 5.2: The operations involved in the global probabilistic inference in the original CNMF network and three different relevant networks.

The relevant network \(B^{(E_1, Q)}\)

Figure 5.4 depicts the directed acyclic graph of \(B^{(E_1, Q)}\). We make the following observations concerning the topology of the directed acyclic graph and the state
Figure 6.3: Directed Acyclic Graph of the complete CMMF-network B.
spaces of some cliques.

The directed acyclic graph possesses a tree structure. This tree contains the variables also contained in focal areas (1) and (4) with the exception of the vertex subset

\{congenital disorder, measured peth., measured val., time lapse(2×), val. to mother, peth. to mother\}
and with the addition of vertex \textit{TraumaNM} – \textit{glyamNM}.

Vertex \textit{TraumaNM} – \textit{glyamNM} has been added to $G^{(E_1,Q)}$ as a result of the moral connection it possesses with \textit{drugsNM}. The addition of vertex \textit{TraumaNM} – \textit{glyamNM} ensures that the junction tree associated to $B^{(E_1,Q)}$ can easily be combined with the junction tree associated to the relevant graphs $B^{(E_2,Q)}$ and $B^{(E_3,Q)}$ to be discussed shortly.

The vertex subset \{\textit{congenital disorder, measured peth., measured val.}\} has been pruned as a result of focusing and the vertex subset \{\textit{val. to mother, peth. to mother}\} has been pruned as a result of evidence absorption.

We note that the contribution from evidence absorption to the gain is obtained because the cliques \{\textit{peth. to mother, time lapse, peth. conc. baby}\} and \{\textit{val. to mother, time lapse, val. conc. baby}\}, both with clique size 36, are not included in the junction tree. We mention that the omission of these cliques results in the saving of 484 operations, which is twice the sum of the clique sizes of clique \{\textit{peth. to mother, time lapse, peth. conc. baby}\} and its neighbor in the junction tree of the focused network, clique \{\textit{peth. cons. baby, drugs, val. cons. baby}\}, plus twice the sum of the clique sizes of clique \{\textit{peth. to mother, time lapse, peth. conc. baby}\}.
and \( \{ \text{peth. cons. baby, drugs, val. cons. baby} \} \). The omission of the cliques accounts for an increase in the gain with respect to the focused network by a factor 1.93.

The relevant network \( B^{(E_2,Q)} \)

Figure 5.6 depicts the directed acyclic graph of \( B^{(E_2,Q)} \). We make the following observations concerning the topology of the directed acyclic graph and the state spaces of the some cliques.

The directed acyclic graph contains the variables also contained in focal area (2) and (4) with the exception of the vertex subset \( \{ \text{water load, blood glucose, NH4, prolonged labor, immaturity} \} \), and with the addition of the vertex subset \( \{ \text{TraumaNM, drugNM, TraumaNM} - \text{glyamNM} \} \).

The vertex subset \( \{ \text{TraumaNM, drugNM} \} \) has been added to ensure that the junction trees associated with \( B^{(E_1,Q)} \) and \( B^{(E_2,Q)} \) can easily be combined with the junction tree associated to \( B^{(E_2,Q)} \). The vertex subset \( \{ \text{glucose infusion, treatment} \} \) has been added to incorporate into the relevant network the combined effects they have with the variables \( \text{hypoglycemia} \) and \( \text{hyperammonimia} \) respectively. The vertex \( \text{TraumaNM} - \text{glyamNM} \) must be viewed as an intermediate. This vertex has been introduced into the model based on complexity issues\(^2\).

The vertex subset \( \{ \text{water load, blood glucose, NH4,} \} \) has been pruned as a result of focusing and the vertex subset \( \{ \text{prolonged labor, immaturity} \} \) has been pruned as a result of evidence absorption. We note that applying evidence absorption in vertex \( \text{immaturity} \) cuts through the (only) loop

\[
\{ \text{immaturity, liver dysfunction, hyperammonimia, glyamNM, hypoglycemia, hypoglycemia tendency, immaturity} \}.
\]

This cut results in a saving of 832 operations.

In addition, the clique \( \{ \text{prolonged labor, immaturity, hypoglycemia tendency} \} \) is not included in the junction tree. We mention that the omission of this clique results in the saving of 144 operations, which is twice the sum of the clique sizes of clique \( \{ \text{prolonged labor, immaturity, hypoglycemia tendency} \} \) and its neigh-

\(^2\)The specification of the combined effects that more that two parent-vertices have on a common child can be separated into the specifications of the combination of couples, provided that the so formed couples allow for an independent combination.
boring clique, \(\{\text{glucose infusion, hypoglycemia tendency, hypoglycemia}\}\) in the junction tree of the focused network.

**The relevant network** \(B^{(E_3,Q)}\)

Figure 5.5 depicts the directed acyclic graph of \(B^{(E_3,Q)}\). We make the following observations concerning the topology of the directed acyclic graph and the state spaces of some cliques.

The directed acyclic graph contains the variables also contained in focal areas (3) and (4) with the exception of vertex set \(\{\text{CT, Trauma Cause, Vitamine K, Steroid}\}\), and with the addition of the vertex subset \(\{\text{drugNM, TraumaNM – glyamNM}\}\). Similarly to the previous discussion, the vertex \(\text{TraumaNM - glyamNM}\) must be viewed as an intermediate.

The vertex subset \(\{\text{Trauma Cause, Vitamine K, Steroid}\}\) has been pruned as a result of evidence absorption. We note that applying evidence absorption to vertex \(\text{Steroid}\) accounts for a saving of 192 operations, which is twice the sum of the clique sizes of clique \(\{\text{Brain Edema, TraumaNM, Trauma Manifest}\}\) and its neighboring clique, \(\{\text{BrainEdema, TraumaNM, TraumaManifest}\}\), in the junction tree of the focused network.

### 5.7.3 Alarm

We discuss the effect of the method of problem directed decomposition on the Alarm network. The directed acyclic graph of the domain model is shown in Figure 6.8. We note the following statistics related to the graph size and topology. The directed acyclic graph contains 37 vertices and 42 edges. A total of 20 variables is contained in one or more loops, thus approximately 54% of the variables are contained in a loop. In contrast to the CNMF model, the loops are not isolated from each other and may contain up to 9 variables. Figure 5.7 shows the multiply connected topology of the graph schematically.

Appendix C contains a specification of the state spaces of the contained variables, a list of abbreviations used, and the description of the meaning of variables representing pathophysiological states.

The inspection of the the model does not immediately provide the identification of a natural partition into submodels. However, to ease the discussion of the example, we divide the model into the following focal areas.
Figure 5.7: The graph topology of the loops contained in the Alarm model.

1. Ventilation

2. Circulation

We stress that there is no need to identify the focal areas in the model before applying the method of problem directed decomposition. In Chapter 4, we put it even more strongly, arguing that the method of problem directed decomposition follows the problem (instance) oriented approach, and that a partitioned of the model in focal areas automatically results given the specific problem instances. The reasons why, in this section, we considered the model structure before the specification of the problem instances is in order to develop an idea about the increase in performance that applying the method of problem directed decomposition can possibly provide.

In the directed acyclic graph shown in Figure 6.8 we distinguish the subgraphs induced by (1)

\{Hypovolemia, LV Failure, History, LVED Volume, CVP, PCWP, Stroke Volume, Anaphylaxis, Insufficient Anesthesia, TPR, Catecholamine, Blood Pressure, CO, Heart Rate, Error Low Output, Error Cauter, HR BP, HR EKG, HR SAT\}

and (2)
We note that the two focal areas overlap in vertices \{\textit{Catecholamine, Insufficient Anesthesia}\}. Further, the two focal areas are not separable by the elimination of a single arc. Stated more precisely, the vertex \textit{Catecholamine} is contained in a loop both in focal area circulation and in focal area ventilation. Based on the previous observation, we can already anticipate that the method of problem directed decomposition will not be able to separate the focal areas due to the presence of the moral connection between the vertices \textit{Insufficient Anesthesia} and \textit{SaCO2}.

Considered problem instances

The problem instances we consider are of the diagnosis type. The variables of interest are as follows.

\{\textit{Hypovolemia, LV Failure, Anaphylaxis, Insufficient Anesthesia, Pulmonary Embolus, Intubation, Kinked Tube, Disconnection}\}

The observable variables are as follows.

\{\textit{CVP, PCWP, TPR, Blood Pressure, CO, HR BP, HR EKG, HR SAT, SaCO2, PAP, FiO2, Pressure, Min. Vol., Machine Volume}\}

It is to be noted that approximately 68\% of all variables are contained in either one of the above-given sets. For purposes of comparison we note that in the CNMF model this percentage is only approximately 27\%.

In order to specify district problem instances we partition the set of variables of interest into three subsets.

\[
Q_1 = \{\text{Insufficient Anesthesia}\}
\]
\[
Q_2 = \{\text{Hypovolemia, LV Failure, Anaphylaxis}\}
\]
\[
Q_3 = \{\text{Pulmonary Embolus, Intubation, Kinked Tube, Disconnection}\}
\]

The query in $Q_1$ relates to both circulation and ventilation. The variables contained in $Q_2$ relate to mechanics of the heart. The variables contained in $Q_3$ relate
to faults in the ventilation machine and the pathophysiological state of the lung diseases with emboli.

The variables contained in the query set $Q_1$ relates to variables in both focal areas, since the biological mechanisms for ventilation and circulation are both affected. We note that the previous statement holds for the majority of the variables contained in the model. This is because of the existence of physiological regulation mechanisms present between the two focal areas. In the model, this extended relativity is reflected by the fact that the vertices Insufficient Anesthesia and Catecholamine are contained in the intersection of both focal areas.

Notwithstanding the relatively dense connectivity of the model, and the corresponding extensive relativity of the variables contained, we defined three distinct sets of evidence variables. We expect, however, to find that the relevant networks associated could well show overlapping vertex sets.

The following evidence sets were specified. The heart rate of the patient is a relatively direct indication for the hypothesis of insufficient anesthesia. We, therefore, considered the following evidence set.

$$E_1 = \{HR,BP,HR,EKG,HR,SAT\}$$

The second evidence set is defined as follows.

$$E_2 = \{CVP,PCWP,Blood\ Pressure,CO,History\}$$

The variables contained in $E_2$ are indicative for the variables contained in $Q_2$, and related to the circulation of the patient.

The third evidence set contains the variables indicative for the variables in $Q_3$, and related to the ventilation of the patient.

$$E_3 = \{Machine\ Vol.,Min.\ Vol.,Exp.\ CO2,Pressure,FiO2\}$$

Results and discussion

The relevant network $B_{(E_1,Q_1)}$

Figure 6.9 depicts the directed acyclic graph of $B_{(E_1,Q_1)}$. We make the following observations concerning the topology of the directed acyclic graph. Firstly, we observe that there is no contribution of evidence absorption to the gain. Secondly, all variables contained in the loops as depicted schematically in Figure 5.7 are
Figure 6.8: Directed Acyclic Graph of the complete Alarm network B.
Figure 6.9: Directed Acyclic Graph of the complete Alarm-network $G^{(E_1,Q_1)}$. 
Figure 6.10: Directed Acyclic Graph of the network $G_{E_2, Q_2}$. 
Figure 6.11: Directed Acyclic Graph of the network $G^{(E_3, Q_3)}$. 
<table>
<thead>
<tr>
<th>Problem directed decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>$B$</td>
</tr>
<tr>
<td>$B^{(E_1, Q)}$</td>
</tr>
<tr>
<td>$B^{(E_2, Q)}$</td>
</tr>
<tr>
<td>$B^{(E_3, Q)}$</td>
</tr>
</tbody>
</table>

Table 5.3: The operations involved in the global probabilistic inference in the original Alarm network and three different relevant networks.

included in the relevant graph, with the exception of the variables contained in the loops that are separable by the elimination of only one link. The set of variables mentioned above is referred to as the coreset. In addition to the coreset, the relevant graph contains the query variables, evidence variables, and the following vertex set:

\[
\{\text{Stroke Volume, Insufficient Anesthesia, Pulmonary Embolus, FiO2, Kinked Tube}\}.
\]

Each variable of the above set is included as a result of a moral connection with a variable contained in the coreset.

The coreset is included in the two other relevant networks as well for the reasons next discussed. As soon as vertex Catecholamine is included in some relevant graph, the vertex Insufficient Anesthesia is included too, because Catecholamine possesses converging connections with Insufficient Anesthesia and a variable contained in relevant graph (TPR, and SaCO2 and Art. CO2). The resulting moral connection induces a communication channel between the two focal areas which is comprised of more that one vertex. As a consequence (Lemma 4.5.1), the original network cannot be decomposed into the focal areas as long as no evidence has arrived for either vertex TPR or for both variables contained in vertex set $\{\text{SaCO2, Art.CO2}\}$.

**The relevant network $B^{(E_2, Q_2)}$**

Figure 6.10 depicts the directed acyclic graph of $B^{(E_2, Q_2)}$. We discuss the topology of the relevant graph shown. The relevant graph contains all variables contained in the coreset plus the query variables, the evidence variables
and the vertices contained in the chains that connect the query variable to a variable from the coreset or the evidence variables. Further, the vertices \{Insufficient Anesthesia, Pulmonary Embolus, FiO2, Kinked Tube\} are included, again as a result of the addition of moral connections.

Evidence absorption has been applied on vertex History and vertex CO. The clique \{History, LV Failure\} has not been included in the junction tree of the relevant graph which amount to the saving of \(4 \times 12 = 48\) operation. In addition, arc (CO, Blood Pressure) has been eliminated. The arc elimination cuts through the loop [TPR, Blood Pressure, CO, Heart Rate, Catecholamine, TPR], resulting in a less complex network and a saving of 332 operations during probabilistic inference. The sum of the effect of applying evidence absorption in this case has resulted a computational effort of 2352 operations corresponding to a gain factor of approximately 1.3.

The relevant network \(B^{(E_3,Q_3)}\)

Figure 6.11 depicts the directed acyclic graph of \(B^{(E_3,Q_3)}\). The relevant graph contains all variables contained in the coreset plus the query variables, the evidence variables and the vertices contained in the chains that connect the query variable to a variable from the coreset or the evidence variables. Furthermore, the following vertex set included again because of the addition of moral connections.

\[
\{\text{Stroke Volume, Insufficient Anesthesia, Pulmonary Embolus, FiO2, Kinked Tube}\}
\]

All variables contained in \(E_3\) are leaves, thus evidence absorption does not contribute to the gain.

5.7.4 Closing comparison

In the previous section, we analyzed the impact of applying the method of problem directed decomposition to two real-life networks, the CNMF model and the Alarm model. The models contain approximately the same number of vertices, 35 and 37 respectively, but differ in their connectivity, the number of arcs is 36 and 42 respectively. In addition, they possess quite different global topological properties. The CNMF model contains two non-overlapping loops of moderate length (approximately 25\% of the variables are contained in a loop), while the possibly
overlapping loops in the Alarm model are more in number and more extended (approximately 54% of the variables are contained in at least one loop).

Based on these model characteristics, we expect that the method of problem-directed decomposition will perform better when applied to the CNMF model than when applied to the Alarm model. This is confirmed by the experiments performed. The CNMF model allows for a separation into submodels of comparable size, while the partition of the Alarm model into submodels is such that each submodel contains at least the the coreset.

In addition, the problem instances specified on the CNMF model all contained the same single query variable, while in the problem instances specified on the Alarm model a total of 8 different query variables had been specified. Coincidently, the size of the evidence sets matched in both sets of experiments. Further, in the CNMF model, both the query variable and the evidence sets were located in the disjunct focal areas which led to a nice decomposition of the model, while in the Alarm model the query and evidence sets located such that in all relevant networks the coreset was included.

Finally, the problem instances specified on the CNMF model were of the prognosis type, meaning that the pieces of evidence are entered toward the roots of the network, and evidence absorption pays a major contribution to the overall gain factor. For the problem instances specified on the Alarm model the opposite was true. The considered problem instances were diagnoses, meaning that the pieces of evidence are entered toward the leaves of the network and evidence absorption contributed only minor to the overall gain factor.
Chapter 6

Decision networks

6.1 Introduction

We recall from Chapter 3 that the Bayesian belief network formalism provides a framework for various reasoning tasks, and that the result of a reasoning task is an updated probability distribution for the (hypothetical) variables conditional on the evidence entered in the network. However, a probability distribution on its own does not provide a means to act. Addressing planning problems is the area of decision theory, and this is concerned with making choices.

In the present chapter, we investigate the connection between Bayesian belief networks and decision theory. We review the various approaches to planning, and arrive at the conclusion that the formalism of decision networks looks promising as a tool to address the planning problem. Decision networks provide a framework for the evaluation of a decision problem. The evaluation of a decision network gives rise to a serious computational problem. However, the specification of the decision problem is recognized to possess a direct relation to a (specific) problem instance, and, therefore, the method of problem-directed decomposition proposed in Chapter 4 can be applied to release the computational effort.

In addition, we report on a comparative study between the decision network formalism and the decision tree formalism. We limited the comparative study to decision trees because our main interest concerns evaluation. The obtained results also apply to the other sketched formalisms since they employ the decision tree formalism in the evaluation process.

This chapter is organized as follows. Section 6.2 sketches related formalisms to incorporate the ability to plan actions in Bayesian belief networks as well as the
decision tree approach. The decision network formalism is discussed in Section 6.3. Section 6.4 reports on a comparative study between a decision network and a decision tree for the single-test/single-treat decision problem. In Section 5.6, the first part of the chapter is concluded by a discussion. The chapter continues with an illustration of the use of problem directed decomposition in planning problems. In Section 6.6, we reconsider the Central Neural Muscular Failure (CNMF) example discussed in the previous chapter. We extend the CNMF network to a decision network and demonstrated the application of the method of problem directed decomposition.

6.2 Approached to the planning of actions

Statistical decision theory emerged with the work of von Neuman and Morgenster [69]. This theory requires that probabilities or chance are based on repeatable chance experiments. For real applications, these frequency based or objective probabilities are often hard, or even impossible, to acquire. In the Bayesian approach to decision theory [17, 50], subjective probabilities are considered as valid as objective ones, opening interesting application areas such as for instance medicine.

Before proceeding, we introduce the terminology used throughout this chapter. The decision problem or planning problem can be formulated as a trade-off between the risks of actions and the expected benefits of these actions. Solving the decision problem consists of finding the best action(s) to perform. In the process, choices between alternative actions must be made. Making these choices requires knowledge about the desirability of the consequences of the actions. The last-mentioned knowledge source is commonly represented by a utility function. Generally, there are multiple actions available, and the decision-maker must be able to make these trade-offs simultaneously such that the optimal action can be chosen. Determining the best choices is called the evaluation of a decision problem.

Decision trees

Traditionally, decision trees have been used to formulate decision problems as the set of all possible chronological sequences of actions or strategies, see for example [70]. On the one hand, listing all strategies allows for a computational simple evaluation. On the other hand, the decision tree representation generates
a 'combinatorial explosion'. As a result, the enormous storage space requirement is one of the major disadvantages. Other well-known problems with decision trees are that the chronological order of actions often require assessment of conditional probabilities which are rarely natural to human experts. Moreover, (conditional) independence of the statistical variables is not visualized in the tree which adds to the difficulty of probability assessment. In fact, the causal knowledge about how events influence each other is not explicitly represented in the decision tree. For this reason, the decision tree representation does not enable the calculation of arbitrary probabilistic queries.

*Bayesian belief networks*

A major development toward parsimonious representation of probability distributions and inference was the formulation of Bayesian belief networks as discussed in Chapter 3. After the introduction of Bayesian belief networks, it was recognized that this parsimonious representation with clearly visualized independence relations could be used to ease the problems mentioned above.

*Influence diagrams*

The first attempt to combine Bayesian belief networks and decision theory resulted in the introduction of influence diagrams [33]. Basically an influence diagram consists of a Bayesian belief network augmented with decision vertices and one value vertex. In an influence diagram, the sequence of permissible decisions is represented by *informational links* which induce a chronological order of decisions. Influence diagrams are shown to be functionally equivalent to decision trees, which established the link to the traditional approach to decision making.

There are two methods of determining the optimal decision policy from an influence diagram. Howard and Matheson's method, [33], which consists of converting the influence diagram into a decision tree and solving the decision problem in the decision tree formalism. The constructed decision tree will contain all possible orderings of decision variables, which results in an enormous storage space requirement as the major disadvantage.

And Shachter's method, [52], which consists of repeated vertex elimination steps combined with arc reversal steps. The vertex elimination may overcome some of the storage space problems, but the arc reversal step may induce excessive
storage space requirements since a large number of arrows may be directed to a particular vertex. In addition, modeling restrictions are necessary to facilitate this method.

Hybrid

To get around the exponentially growing storage requirement, a hybrid method was suggested by Pearl [47]. The method is based on the realization that the decision tree need not actually be stored in totality to enable its evaluation. The method employs an influence diagram to dictate the permissible sequences of actions. During the evaluation, a Bayesian belief network that contains all chance variables (with their causal interactions) is used to calculate the probabilistic parameters necessary for the calculation of the utility function. An exhaustive search of all of permissible strategies is required to solve the problem. However, only the path actually under evaluation needs to be stored.

Summarizing, it can be stated that the influence diagram offers a solution to the problem of unnatural probabilistic assessments that are often required in decision trees. Further, the hybrid method offers a solution to the problem of storage requirements that occurs during the evaluation of an influence diagram. We note that in both the influence diagram and the hybrid method the modeling of domain knowledge and knowledge about permissible strategies is mixed. In addition, only predefined strategies are evaluated.

Decision networks

Recently, Andreassen proposed a method to solve the decision problem in extended Bayesian belief networks [1]. It has been demonstrated that the problems mentioned above do not exist in the decision network formalism [65, 63]. In addition, decision networks contrast with the approaches mentioned, in that the permissible sequences of actions are not predefined. The (causal) domain knowledge and the knowledge about decision strategies are not mixed. Knowledge about decision strategies is, in fact, not represented explicitly. A strategy is constructed dynamically during the evaluation process. The decision problem is to choose, repeatedly, only the next action. At any decision point, a decision-maker can choose from the full range of actions available. The choice is made in the light
of the present evidence, which provides for flexible planning tuned to the specific
decision problem at hand.

\textit{Differences and resemblances}

An influence diagram and a decision network are essentially the same repre-
sentation of a decision problem in the sense that they contain the same decision
nodes and chance nodes. There are two differences. Firstly, the utility nodes in
a decision network are merged together in one \textit{value node} in a influence diagram.
This value node receives links from all attributes that together determine the
expected utility. The value node can be considered as a look-up table which con-
tains for every possible configuration of attributes a utility. The \textit{expected} utility
for a specific configuration (associated to a specific strategy) is calculated as the
weighted average of the utilities of that configuration where the weighting factors
are the probabilities of the uncertain attributes. Secondly, an influence diagram
possesses \textit{informational links}. These links enter decision nodes and show which
variable should be known to the decision-maker before the decision can be made,
they imply time precedence. The freedom of total ordering of decision nodes in
the decision network formalism allows flexible planning dedicated to the specific
decision problem at hand. We mention that, in the decision network formalism, a
decision variable may not even receive any links, this means that the value of any
individual decision is computed without the need for a predefined set of variables
(decisions) already known.

Regarding the construction of the different graphical models we make the
following observation. The constructor of a decision tree formulates the deci-
sion problem in a set of ordered sequences of actions, implicitly using domain
knowledge to guide this task. The construction of a influence diagram relies on
knowledge about strategies as they are represented by the informational links.
In contrast, building a decision network is primary concerned with modeling the
domain knowledge; no sequences of actions are predefined. Decision support will
emerge from the model while evaluating the decision network. For this reason we
expect it is easier to explain decision support behavior in the decision network
approach than in the decision tree approach, because plausible motivation of the
suggested optimal strategy requires access to the implicate domain model which
is not actually available in the decision tree approach.
6.3 Decision networks

6.3.1 From Bayesian belief network to decision network

A decision network is an extended Bayesian belief network [1]. The extension consists of the addition of decision vertices that represent actions. Decision vertices are roots in the graph but do not need probability assessment functions because they are always instantiated by the decision-maker. Furthermore, we need to quantify the seriousness of the consequences of actions. We, therefore, need the ability to assign utilities to vertices that induce risk, the utility vertices. Utility vertices are ordinary chance vertices with attached utility vectors. A Bayesian belief network extended as described above is called a decision network. Formally, a decision network is defined as follows.

Definition 6.3.1 Let $B = (G, \mathcal{P})$ be a Bayesian belief network as defined in Definition 3.2.2. We define $D = (B, (D, U), \mathcal{U})$ to be a decision network associated to $B$ where,

- $D \subseteq V(G)$ is the set of decision vertices;
- $U \subseteq V(G)$ is the set of utility vertices, such that
  $$D \cap U = \emptyset$$
- $\mathcal{U}$ is the set of utility vectors.

The utility vector of a utility vertex $U_i \in \mathcal{U}$ is denoted by $U(U_i)$, and contains for each state $u_{i,j}$ of $U_i$ a quantification of the risk incurred if the vertex takes that state.

6.3.2 Evaluating decision networks

A decision network supplies decision support in an iterative fashion. In every iteration, the planning of actions proceeds until and including the first information-gathering action. Upon acquiring the information, the associated action is performed. Gathered information and knowledge about the performed actions is fed into the decision network and their impact is propagated throughout the network. The next iteration involves a re-evaluation of the available actions, which is based on a decision network in which the knowledge of the actual observed value of the gathered information and the performed action(s) has been accounted for. In
6.3 Decision networks

![Diagram](image)

**a:** Decision network with two treatment decisions, $D_1$ and $D_2$.

**b:** Decision network with one treatment decision, $D_2$, and one test decision, $T_1$.

Figure 6.1: The graphical part of the decision networks used in the examples.

In principle, the complete set of available actions is submitted to the re-evaluation. Consequently, the same action can be performed repeatedly.

In the following discussion we assume, without loss of generality, that all variables are binary. We denote a single treatment decision with $D_i$ and we denote a single test decision with $T_i$. For example, $Sp(D_1) = (d_1, \neg d_1)$, $Sp(U_1) = (u_1, \neg u_1)$, $Sp(C_1) = (c_1, \neg c_1)$. The decision variable $D_i$ can take one of two states: $d_i$ denoting act and $\neg d_i$ denoting do not act.

We discuss the evaluation process using the example decision networks depicted in Figure 6.1.a and b. Decision vertices are represented by rectangle frames, $\{D_1, D_2\}$, utility vertices are represented by rounded frames, $\{U_1, U_2\}$ and the remaining change vertices are represented by circular frames, $\{C_1, C_2, R_{T_1}\}$. The decision network is completed with the set of (conditional) probability tables $\mathcal{P}$ and a set of utility vectors $\mathcal{U}$. For the decision network depicted in Figure 6.1.a we have for example:

$$\mathcal{P} = \{ P(C_1), P(C_2 \mid D_1 \land D_2 \land C_1), P(U_1 \mid D_1), P(U_2 \mid C_2) \}$$

and

$$\mathcal{U} = \{ U(U_1), U(U_2) \}$$

The probability assessment functions in $\mathcal{P}$ specify the joint probability distribution on the variables $C_1, C_2, U_1, U_2$, conditional on the variables $D_1$ and $D_2$. The vectors in $\mathcal{U}$ specify the utilities in the decision problem. To find the action that
maximizes the utility function, the utility function must be calculated for each element of \( C^*_D = \{ (d_1, d_2), (d_1, -d_2), (-d_1, d_2), (-d_1, -d_2) \} \).

Two types of action have to be distinguished. Referring to a medical application domain we speak of test actions and treat actions. In Figure 6.1.a, \( D_1 \) and \( D_2 \) represent treat actions. We first discuss planning treat actions.

**Treat actions**

As can be seen from Figure 6.1.a, treat actions affect chance variables. This effect is calculated by a single propagation through the decision network. The consequences of actions that should be accounted for are represented by utility vertices. The goal of the planning process is to maximize the utility function. The utility function of a treatment decision is calculated as described below.

Suppose that the utility function for the combined treatments \((d_1, d_2)\) must be determined. This utility function is denoted with \( F_{tr}(d_1, d_2) \). The instantiations \( D_1 = d_1 \) and \( D_2 = d_2 \) are fed into the decision network and propagated through the network to achieve the updated probability distributions of the utility vertices.

Subsequently, the utility function is calculated as the sum of the value functions associated to the utility vertices. Each value function is calculated local to the utility vertex as the weighted sum of the utility vector where the weighting factors are the corresponding elements of the updated probability distribution of this vertex. In the example, for the value function for the utility vertex \( U_i \) we get the following.

\[
V^{(d_1,d_2)}(U_i) = \sum_{u_{i,j} \in S_F(U_i)} P(u_{i,j} \mid d_1 \land d_2) \ast U(u_{i,j})
\]

The superscript \((d_1,d_2)\) indicates the assumed actions. The notation of the updated probability, as calculated in the network, has been augmented with a conditional part to indicate the current instantiations. For example \( P(U_i \mid d_1 \land d_2) \) denoted the updated probability of vertex \( U_i \) after \( D_1 = d_1 \) and \( D_2 = d_2 \) have been propagated through the network.

The (treatment) utility function is then calculated as follows.

\[
F_{tr}(d_1, d_2) = \sum_{U_i \in U} V^{(d_1,d_2)}(U_i)
\]

The optimal treatment decision, denoted with \( c^*_t \), is found by searching \( C^*_D \) for the one that maximizes the utility function:
\[ c^*_t = \max_{c^*_D \in c^*_D} F(c^*_t) \]  \hspace{1cm} (6.1)

**Test/treat actions**

Figure 6.1.b depicts one treat action, \( D_2 \), and one test action \( T_1 \). After a test action has been taken, the test result, \( R_{T_1} \), becomes available.

Suppose that \( T_1 \) is a test decision with \( Sp(T_1) = (do\ T_1, do\ not\ T_1) \), or \( Sp(T_1) = (t_1, \neg t_1) \). Performing the test will lead to a test result, \( R_{T_1} \). Suppose further that this test result can be either positive, \( pos \), or negative, \( neg \), i.e. \( Sp(R_{T_1}) = (pos, neg) \).

The decision-maker can decide to perform a test but has to wait for nature to instantiate the test result. The dashed arc indicates the time lapse between the decision to test and the availability of the test result. Obviously, the decision-maker cannot wait till the test result has become available and therefore shall base his decision on the *expected* value of the test.

The (expected) utility function of performing the test, also called test utility function, and denoted with \( F_t(t_1) \), is calculated as follows. For each possible test result, the value is instantiated and propagated through the network. Subsequently, the optimal configuration of the treat decisions is determined and the associated utility function is calculated.

The instantiation of a positive test result, for example, yields the following treatment utility function.

\[ F^{R_{T_1}=pos}_{tr}(c^1) \]

\( c^1 \) denotes the optimal treatment decision following a positive test result, and is calculated using Equation 6.1. The superscript \( R_{T_1}=pos \) is added to the treatment utility function to indicate the instantiated test result.

The utility function of performing the test is then calculated as the weighted average of the treatment utility functions, where the weighting factors are the probabilities that the associated test result is actually found.

\[ F_t(t_1) = \frac{F^{R_{T_1}=pos}_{tr}(c^1) \times P(R_{T_1} = pos)}{F^{R_{T_1}=pos}_{tr}(c^1) \times P(R_{T_1} = pos) + F^{R_{T_1}=neg}_{tr}(c^2) \times P(R_{T_1} = neg)} \]  \hspace{1cm} (6.2)

In the equation above, \( c^2 \) denoted optimal treatment decision following a negative test result. The test is performed if the value of the test utility function exceeds the
value of not performing the test. The computation of this last value is analogous to the computation of the optimal treat decision(s), with $T_1 = \neg t_1$ instantiated. Thus the value of the information acquired by performing the test $T_1$ is calculated as follows.

$$F_1(t_1) - F_r^{t_1}(c_{tr}^*)$$ (6.3)

Generally, when there are multiple tests available, say $\{T_1, T_2, \ldots, T_m\}$, the best (most informative) test(s) must be found. To this end, in principle, the value for each conceivable combination of tests is determined. The optimal set of tests is found by searching $C(T_1, T_2, \ldots, T_m)$ for the combination, $c_t^*$ and $c_{tr}^*$, that maximizes Equation 6.3.

![Search tree for the evaluation of a decision network comprised of two treatments, $TR_1, TR_2$, and two tests $T_1, T_2$.](image)

Figure 6.2: Search tree for the evaluation of a decision network comprised of two treatments, $TR_1, TR_2$, and two tests $T_1, T_2$.

### 6.3.3 Computational complexity

The evaluation of one iteration of the decision problem with two test decisions, $\{T_1, T_2\}$, and two treatment decisions, $\{TR_1, TR_2\}$, can be viewed as a depth-first
search through the tree depicted in Figure 6.2. Again the decision variables are assumed to be binary. As can be seen from Figure 6.2, a test decision introduces a chance variable with a branch for each test result. Each branch leads to the root of the separate tree representing all possible treat actions.

During the evaluation of a decision network, one path from the root to a leaf is constructed dynamically. The utility function is calculated for the associated decisions. Subsequently, the next path, that differs in only one action from the previous one, is considered. Again, the utility function is calculated and a comparison takes place. The decisions associated with the highest utility function are considered further in the evaluation. This process is repeated until all possible decisions have been considered.

We emphasize that for decision problems of realistic size, the complete evaluation as sketched above is intractable since the number of paths in the tree increases exponentially with the number of decision variables. We, further, stress that the exponential computational complexity is inherent to the complete evaluation of a decision problem, and continue to discuss heuristic evaluation methods. After the heuristic methods have been outlined, we demonstrate the role of problem-directed decomposition in combination with the heuristics.

Myopic planning

A common solution to the evaluation problem is pruning the tree shown in Figure 6.2. The pruning limits the search space for the optimal next decision. Normally, the pruning results in a tree where only a single action may be performed at a time. The following heuristic rules underly this pruning technique.

1. Each information source is evaluation in isolation, as if it were the only information source available.

2. After the information source has been consulted, only one action may be performed.

Applying the two rules listed above results in the myopic approach to the selection of information sources. In our example, the myopic approach would result in the pruning of the subtree rooted at vertex $R_{T1} \land R_{T2}$ as well as the uppermost branch of the "TREAT" subtree. Obviously, the myopic approach to planning leads to suboptimal solutions. It, then, depends on the application as to whether or not the suboptimal solutions are acceptable. We note that in myopic
planning, all hypotheses participate during the evaluation process. That is, the generalized control strategy is followed in order to decide which information source to consult next.

**Planning with subgoals**

The computationally more efficient control strategies that divide the decision problem into subproblems, each establishing a limited context for the evaluation process, is called subgoaling.

A subgoal may correspond to a subset of all hypotheses. In this case, focusing on a subgoal normally reduces the computational effort involved in the evaluation process, since it narrows the set of information sources from which the best one must be selected. A subgoal may also correspond to a sub-planning problem. Again, focusing normally results in a reduction of the computational effort involved in the planning process, since only a subset of the decision variables are considered in the evaluation process.

When subgoaling is applied, a distributed control strategy is responsible for the evaluation of the subgoal, as well as for the adjustment of the subgoal as a consequence of new information. The adjustment may consist of the extension of the subgoal, or a shift of attention to another subgoal.

In a hypothesis-driven control process, the decision network could be used to determine the most probable explanation for the present findings, in order to guide the identification of the (current) subgoal. In an evaluation-driven control process, the progress of the evaluation process determines the current subgoal. An example of the latter control in subgoaling, in connection with the method of problem-directed decomposition, is discussed in Section 6.6. Finally, we mention that, besides the computational advantages, subgoaling has an additional psychological advantage as it prevents abrupt shifts of attention to seemingly unrelated topics.

**Problem instance mapping**

A natural way to address the computational problem inherent to the probabilistic inferences required during the evaluation of decision networks is applying the method of problem-directed decomposition as proposed in Chapter 4. The propagation of the impact of the actions through the whole decision networks, a part of the evaluation of a single path, can be restricted to the utility variables. The propagation part of the evaluation, then, is formulated as solving a problem instance.
The set of decision variables and test results corresponds to the evidence set \( E \), and the set of utility variables corresponds to the query set \( Q \). The propagation of the impact of action and gathered information is performed in the relevant decision network.

Additionally, subgoals can be defined in terms of problem instances in cases where the control strategy of subgoaling is utilized. The subgoals, represented by problem instances, give rise to separated decision networks. The evaluation of the separate decision networks might be of considerably smaller complexity than the evaluation of complete (relevant) decision network.

### 6.4 Decision networks and decision trees

To gain insight in the decision network approach, a comparison with the decision tree approach is made in this section.

To this end, we analyze the evaluation of the single-test/single treat decision problem. Since only one treat decision and only one test decision are available, the evaluation of the decision network is comprised of only one iteration. The result of the decision networks evaluation is either to treat or to wait, or to perform the test, followed by treatment according to the test result. The decision tree captures the three strategies which correspond exactly to the actions mention above. The correspondence enables a comparison of the evaluation processes. We emphasize that the dynamic nature of planning in (general) decision networks is not subject to the comparison.

In the analysis of the single-test/single treat decision problem, a medical context was chosen. The decision problem is formulated as follows. The decision agent suspects a disease \( D \) to be present, and has to decide between three strategies: ‘wait’, ‘test and act according to the test result (R)’ and ‘treat immediately’. Performing the test might result in the manifestation of risk-bearing complications (CP). Administering the treatment might, on the one hand, induce risk-bearing side effects (SE). On the other hand, the probability of the presence of the disease in the future, disease continued (DC) will diminish. All chance and utility variables are binary: \( Sp(D) = (d, \neg d) \), \( Sp(SE) = (se, \neg se) \), \( Sp(CP) = (cp, \neg cp) \), \( Sp(R) = (r, \neg r) \), \( Sp(DC) = (dc, \neg dc) \). The decision variable ‘test’, \( T \), can be ‘test’, \( t \), or ‘do not test’, \( \neg t \), and the decision variable ‘treat’, \( TR \), can be ‘treat’, \( tr \), or ‘do not treat’, \( \neg tr \). Figure 6.4 shows the structure of the decision tree with three strategies: ‘wait’, ‘test’ and ‘treat’. Figure 6.3 depicts the structure of the
decision network for the same decision problem.

![Decision network diagram](#)

Figure 6.3: The single-test/single-treat decision network.

![Decision tree diagram](#)

Figure 6.4: Decision tree for the single-test/single-treat decision problem.

In Section 6.4.1, we discuss the decision network for the single-test/single-treat decision problem. The decision tree approach for the single-test/single-treat decision problem is discussed in Section 6.4.2.

In the single-test/single-treat decision problem there is one parameter that varies with the particular decision problem. This parameter is the prevalence of the disease, \( P(d) \). To study the differences in (optimal) strategy or action
between the decision tree and the decision network, for varying prevalence of the disease, two sensitivity analyzes were conducted and presented in Section 6.4.3. The results of these analyzes are compared in Section 6.4.4.

6.4.1 The single-test/single-treat decision network

The logic of the single-test/single-treat decision problem is reflected in the topology of the decision network shown in Figure 6.3.

Consider the left part of the decision network: SE (side effects), TR (treat), and DC (disease continued). On one hand, administering a drug could have the negative effect of side effects, but, on the other hand, the positive effect of reducing the likelihood of disease continued. Further, the manifestation of side effects does not influence the positive effect of the drug. That is SE and DC are conditional independent given TR.

Next, consider the middle part: DC, D and R (result). The probability distribution of the test result R depends on the prevalence of the disease. After a test result has been found it will change the clinical likelihood of this disease. The arc from D to R, represents this dependency, and is quantified with the sensitivity and the specificity of the test. The dashed arc from T to R, indicates that a test result will become available after the test has been performed. We note that this link is conceptually not part of the the decision network.

The strength of the causal relations are quantified by conditional probabilities as follows.

\[ P(SE \mid TR), P(DC \mid TR \land D), P(R \mid D), P(CP \mid T) \]

The prevalence of the disease is expressed by the prior probability \( P(d) \). The decision vertices, treatment and test need no quantification. They are always instantiated.

The calculation of the utility function requires a quantification of the risk incurred by the utility vertices. The following utility vectors are specified.

\[ U(CP) = (0,1), \ U(SE) = (0,1), \ U(DC) = (0,1) \]

6.4.2 The single-test/single-treat decision tree

Consider the decision tree depicted in Figure 6.4. The rectangular node labeled X represents the decision between the three strategies. The rectangular node
labeled TR represents the decision to treat. The circular chance nodes: SE, CP, D and DC represent side effects, complications, disease and disease continued respectively.

The links that sprout from a decision represent the actions available at that particular decision point. The links that sprout from a chance node are quantified with the probability of the associated state. In cases where a node representing a test result precedes the chance node in the strategy, the probability of this chance node is updated with the effect the test result exercises on the associated chance event. The calculations involved in this updating process are carried out separately from the decision tree. In the other cases, the probabilities are directly accessible, see Table 6.1.

Consider Figure 6.4, the link labeled (a) is quantified with $P(\neg d)$. The link labeled (b) is quantified with $\frac{P(\neg d) + P(\neg d) \cdot P(pos)}{P(pos)}$, that is $P(\neg d)$ updated in the light of a positive test result. Likewise, the link labeled (c) is quantified with $\frac{P(\neg d) + P(\neg d) \cdot P(neg)}{P(neg)}$.

The dots: ... indicate that the right-hand side of the portion of the decision tree at the alternative branch is repeated.

The desirability of the outcome of a strategy is quantified with utilities. In the decision tree depicted in Figure 6.4, the outcome of, for example, the treat strategy is side effects, or disease continued, or not disease continued. Both side effects and disease continued have utility 0, expressing that they represent the worst outcome. The best outcome, no disease continued, is quantified with utility 1.

**Computation of the optimal strategy**

With all the information displayed in the decision tree shown in Figure 6.4, we can calculate the utility function for a strategy. This calculation process is called averaging out and folding back [70]. The calculation starts at the leaves of the tree. Working back toward the root, two operations are performed.

Averaging out is the calculation of the expected utility of a chance node. This is the vector product of the probabilities of the links sprouting from that chance node and the numerical expressing of the outcome associated to the endpoint of the link. The expected utility represents the ‘outcome’ for the node to its left and is stored at the chance node such that this process can be repeated in a recursive way.

For example, in the tree shown in Figure 6.4, the expected utility of the up-
permost DC chance node is calculated as:

$$EU(DC) = 0 \times P(dc \mid d \land tr) + 1 \times P(\neg dc \mid d \land tr)$$

Folding back prunes the inferior choices. For every decision vertex, the action which maximizes the expected utility is chosen. This maximal expected utility is stored at the decision variable. All other possible actions available at this decision point, together with the associated subtrees, are pruned from the tree.

### 6.4.3 Analysis

In the sensitivity analysis, the numerical assessment of the probabilities as shown in Table 6.1 were used. We note that all variables are binary and that $tr = pos$ is abbreviated $pos$ and likewise $t = neg$ is abbreviated $neg$. In addition, we note that the correctness and precision of the assumed probabilities do not influence the value of the comparison.

<table>
<thead>
<tr>
<th><strong>Probability of</strong></th>
<th><strong>Assumed value</strong></th>
<th><strong>Formula</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>no disease continued</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>with disease, treated</td>
<td>67</td>
<td>$P(\neg dc \mid d \land tr)$</td>
</tr>
<tr>
<td>with disease, untreated</td>
<td>29</td>
<td>$P(\neg dc \mid d \land \neg tr)$</td>
</tr>
<tr>
<td>with non-disease, untreated</td>
<td>82</td>
<td>$P(\neg dc \mid \neg d \land \neg tr)$</td>
</tr>
<tr>
<td>with non-disease, treated</td>
<td>82</td>
<td>$P(\neg dc \mid \neg d \land tr)$</td>
</tr>
<tr>
<td>side effect from the drug</td>
<td>5</td>
<td>$P(se \mid tr)$</td>
</tr>
<tr>
<td>complications from the test</td>
<td>1</td>
<td>$P(cp \mid t)$</td>
</tr>
<tr>
<td>Positive test result</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In disease (sensitivity)</td>
<td>90</td>
<td>$P(pos \mid d)$</td>
</tr>
<tr>
<td>Negative test result</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In non-disease (specificity)</td>
<td>98</td>
<td>$P(neg \mid \neg d)$</td>
</tr>
</tbody>
</table>

Table 6.1: The second column contains the assumed probabilities in the analysis. The third column contains the formulas.
Figure 6.5: For a clinical likelihood of disease $\leq 0.03$ the best strategy is to wait, for a clinical likelihood of disease in the range $[0.03, 0.44]$ test is the optimal strategy, and for a clinical likelihood of disease $> 0.44$ therapy without testing is optimal.

6.4.4 Comparison

Displaying the utility function per strategy for varying probabilities of disease reveals the graphs shown in Figure 6.5 and Figure 6.6. The visual inspection of the graphs reveals two major differences, a scaling factor between the y-axes, and a bending in the test strategy for the decision network. First, we explain the bending. As mentioned earlier, in the decision network approach, no fixed set of possible strategies needs to be specified. The action to wait after a negative test result can, therefore, change into the action to treat after observing a negative test result. This occurs after the bend in the test strategy. After the bending, treatment will always be justified, and the line continues parallel to the treat strategy. As long as the bending takes place after the intersection with the treat line, the optimal strategy does not change. It can easily been proved that this is the case, provided that no new evidence about the decision problem is entered into the network. In Appendix D we prove the above statement for our example.

To explain the scaling, we recall that in the decision network approach the utility function is calculated as the sum of the values of the utility vertices. Consider for example the treat strategy$^1$, $F(\text{treat}) = P(\neg se | tr) + P(\neg cp | \neg tr) + V^{\text{treat}}(DC)$. The utility function of the treat strategy in the decision

$^1$The differences and similarities between the other strategies, can be explained analogously.
Figure 6.6: For a clinical likelihood of disease \( \leq 0.04 \) the best strategy is to wait, for a clinical likelihood of disease in the range \([0.04, 0.47]\) test is the optimal strategy, and for a clinical likelihood of disease \( > 0.47 \) therapy without testing is optimal. For a clinical likelihood of 0.59 the 'test' strategy shows a bend. The line continues parallel to the 'treat' strategy with a shift of 0.01 due to the loss of utility as a consequence of performing the test.

te tree approach is calculated as the probability of the best outcome, that is, 
\[
P(\neg dc \mid tr) = P(\neg se \mid tr) \ast (P(\neg dc \mid d \land tr) \ast P(d) + P(\neg dc \mid \neg d \land tr) \ast P(\neg d))
\]
(see the tree shown in Figure 6.4). The second factor of the last equation equals \( V^{treat}(DC) \). In our example, both \( P(\neg se \mid tr) \) and \( P(\neg cp \mid t) \) are close to 1. The addition of \( P(\neg se \mid tr) \) and \( P(\neg cp \mid \neg t) \), therefore, causes the shift of approximately 2.

The differences between the intervals where the test strategy is optimal, \([0.04, 0.47]\) for the decision network and \([0.03, 0.44]\) for the decision tree, can be explained by the multiplication factor \( P(\neg se \mid tr) \). Analogously, for the test strategy, the multiplication factors \( P(\neg cp \mid t) \) and \( P(\neg se \mid tr) \) and \( P(\neg se \mid \neg tr) \) are responsible for the shift in the intervals where testing is optimal. The multiplication factors are introduced during the average out and folding back procedure, in order to calculate the weighted probability of the outcome of a strategy. We stress that, in the average out and folding back procedures, the contribution of risk-bearing situations are combined dependently, while in the evaluation of a decision network, the contribution of risk-bearing situations are combined independently. Further research is needed to elucidate the effects of the difference in the calculation of the utility functions for the decision network, and the decision tree
approach for varying (conditional) probability distributions and varying utility assessments. To conclude we would like to mention that the functional equivalence between the decision tree approach and the decision network approach can be forced by the introduction of a “value node” analogous to the influence diagram formalism. We prefer this “value node” to be represented outside the decision network in the control mechanism in order to leave the (causal) knowledge represented in the decision network intact. Clearly, the utility vertices would lose their vectors as only their probability distributions become inputs for the control mechanism.

6.5 Discussion and conclusion

The decision network formalism possesses advantages over decision trees, influence diagrams and the hybrid method. These are the parsimonious representation and the flexible case-specific planning process. The main difference in representation that allows for these advantages is the omission to specify the permissible sequences of actions.

During decision network construction, attention remains focused on the issues of domain modeling and is not obscured by constructing strategies, whereas during evaluation, the logical structure of the network remains untouched and therefore can serve other purposes such as explanation or causal reasoning. The representation of a decision variable as a root vertex prevents problems inherent to the propagation of influence of actions to predecessors of the decision vertex. The conservative planning of actions, up to and including the first information-gathering action, enables the accounting for new information as soon as it becomes available, thereby tuning the planning to the specific case at hand. This flexibility, however, demands an increased computational complexity, because each iteration receives a separate evaluation.

The decision network representation allows for the assessment of utilities as a function of a selected group of variables, possibly containing one variable only. These groups are selected such that the assessment of utility vectors appears most natural to the human expert. The utilities associated to separate groups are assessed independently\(^2\). In addition, the corresponding values are combined independently in order to calculate the utility function. The decision network

\(^2\)We note that the utility vectors must be calibrated in such a way that equally serious situations must be quantified equally.
is used to calculate the updated probability distributions of the utility variables needed in the calculation of the utility function.

Although the presented single-test/single-treat example is very simple, difficulties with probability assessment, as a result of unvisualized independence relationships between the variables in the domain, become apparent. Furthermore, the fundamental difference in storage requirement is clearly illustrated, even for this small example.

The relation between the evaluation of a single iteration in a decision network and the evaluation of decision trees has been investigated. It has been demonstrated that the evaluation in a decision network of (sequences of) actions yields results comparable with the result obtained by the evaluation of the (corresponding) decision tree, provided that no new evidence is entered into the decision network.

In a situation where actions must constantly be tuned to incoming data about the actual situation, the evaluation process must be able to take into account seemingly inferior sequences of action. Failing to take advantage of information as soon as it becomes available can substantially limit the value of planning, since the evaluation of the best action can be based on outdated information. The decision networks approach aims at tuning the planning to the information about the decision problem immediately. A major problem, however, is the resulting computational complexity.

### 6.6 The CNMF decision network

In this section we illustrate the use of problem-directed decomposition in relation to decision network evaluation. To this end, we reconsider the CNMF Bayesian belief network depicted in Figure 6.7 and extend this network to a decision network.

#### 6.6.1 Extension to the decision network

The model has been extended by the addition of the following utility vertices.

\[
\{ \text{complications bs., complications a.v., complications CT, side effects Steroid, side effects treat} \}
\]
The added utility vertices represent binary variables. We distinguish the following subsets of vertices. Treatment decisions.

\{art. ventilation, glucose infusion, treatment, Vitamin K, Steroid\}

Test decisions

\{blood sample, CT\}

Utility vertices.

\{water load, complications CT, complications a.v., side effects Steroid, side effects treat, complications bs., central cyanosis\}

Finally, the added vertices are connected as shown in Figure ??, and quantified with the following conditional probability distributions.

\{P_{CT \ result}(CT \ result \mid Trauma\ NM), P_{complication \ ct}(complication \ CT \mid CT), P_{complication \ bs.}(complication \ bs. \mid blood \ sample)\}

We note that taking a blood sample from the baby may result in values for all variables measurable from the blood. Obviously, pethidine and valium concentrations can only be measured in cases where the drugs have been administered to the mother. The control mechanism takes advantage of this common-sense knowledge in order to guide the evaluation of the value of taking a blood sample. When no drugs are administered, $Sp(HN4) \times Sp(blood \ glucose) = 9$ combinations of test results must be evaluated. This amounts to the evaluation of the best treatment for the 9 different cases. Thus 9 evaluations comparable to the evaluation of a treatment decision must be carried out. We note that the joint probability $P(measured \ blood \ glucose, measured \ NH4)$ is calculated as the product of the individual probabilities since the test results are conditional independent. In addition, we note that the subnetworks created by cutting through the links (drugsNM, CNMFailure), (glyamNM, TraumaNM – glyamNM) and (TraumaNM, TraumaNM – glyamNM) are not d-separated due to the instantiation of vertex art. ventilation.
Figure 6.7: Directed Acyclic Graph of the CNMF decision network.
6.6.2 Problem instance mapping and problem-directed decomposition

Direct problem instance mapping

The problem instance associated to the evaluation problem using a generalized control strategy is as follow.

\[
E = \{ \text{blood sample, measured peth., measured val., blood glucose, NH4, CT, CT result, art. ventilation, glucose infusion, treatment, Vitamin K, Steroid} \}
\]

\[
Q = \{ \text{water load, complications CT, complications a.v., side effects Steroid, side effects treat, complications bs., central cyanosis} \}
\]

Given the above-specified problem instance, the relevant decision network obtained after applying problem-directed decomposition shows the following differences with the original decision network. Evidence absorption has eliminated the outgoing arcs of the evidence vertices. All utility vertices, except central cyanosis, have become one vertex components the values of which are completely defined by their parents. And only one vertex, congenital disorder, has become irrelevant as a result of focusing. During the non-myopic evaluation process, the following number of propagations through the relevant decision network are needed.

\[
\text{Sp(meaured peth.)} \times \text{Sp(measured val.)} \times \text{Sp(blood glucose)} \times \text{Sp(NH4)} \times \\
\text{Sp(CT result)} \times \text{Sp(treatment)} \times \text{Sp(Vitamin K)} \times \text{Sp(glucose infusion)} + \\
\text{Sp(treatment)} \times \text{Sp(Vitamin K)} \times \text{Sp(glucose infusion)} = \\
3^4 \times 2^5 = 2592
\]

Adopting the myopic control strategy reduces the number of propagations to \((2 + 2 + 2 + 2) \times (2 + 3 + 3 + 3 + 3) = 112\).

Problem instance mapping and subgoaling

Next, we discuss a way to apply subgoaling. We assume that the control strategy is represented outside the decision network and discuss the separate evaluations performed in the decision network. We consider the subgoal of the separate planning of treatment actions. We note that the evaluation process will always include the separate consideration of treatment actions. In addition, during the planning of test actions, the optimal treatment decision associated must also be calculated,
albeit for differing (updated) probability distributions. Therefore, a relevant decision network that includes the treatment decision and some of the utility vertices can be used as a starting point for the evaluation of the decision network. In addition to the decision variables, the states of the variables contained in the following set are known.

$$E_0 = \{ \text{immaturity, prolonged labor, peth. to mother,} \$$
$$\text{val. to mother, time lapse(2x), Trauma Cause} \}$$

The problem instance that relates to planning of treatments will be referred to as TreatmentOnly, and is defined as follows.

$$E_1 = \{ E_0 \cup \text{art. ventilation,} \$$
$$\text{glucose infusion, treatment, Vitamin K, Steroid} \}$$

$$Q_1 = \{ \text{water load, complications a.v.,} \$$
$$\text{side effects Steroid, side effects treat, central cyanosis} \}$$

The directed acyclic graph of the relevant decision network $D(E_1, Q_1)$ is shown in Figure 6.8.

**Evaluation**

The decision network $D(E_1, Q_1)$ is used in the evaluation of the treatment decision problem. The following utility vertices are completely defined by their parents.

$$\{ \text{water load, complications a.v., side effects Steroid, side effects treat} \}$$

In order to calculate the updated probability distribution of the remaining utility vertex, central cyanosis, it suffices to call CollectEvidence in the clique that contains the vertex central cyanosis. Note that, in this case, only half of the operations performed in a global propagation are necessary because only the variable of interest is contained in one clique. In the non-myopic evaluation strategy, a total of $2^4 = 16$ probabilistic inferences are performed in the relevant decision network depicted in Figure 6.8, while using the myopic evaluation heuristic, 8 probabilistic inferences suffice.

After the optimal treatment policy has been determined, the process proceeds with the evaluation of the values of the information acquired by performing the tests. The initial subgoal, TreatmentOnly, must now be extended to cope
Figure 6.8: The directed acyclic graph of the relevant decision network $D^{(E_1,Q_1)}$. 
with the determination of the values of the test decisions. The decision problem contains two tests, CT and blood sample. We consider the (combined) test actions: (perform CT scan, take blood sample), (take blood sample only), and (perform CT scan only). Accordingly, we extend the TreatmentOnly subgoal to include

- the CT scan, giving rise to the following problem instance.

\[(E_2, Q_2) = (\{E_1 \cup CT, CT\ result\}, \{Q_1 \cup complications\ CT\})\];

- the blood sample, giving rise the following problem instances.

  - Drug(s) administered to the mother.

\[(E_3, Q_3) = (\{E_1 \cup blood\ sample, measured\ peth., measured\ val.,
blood\ glucose, NH4\}, \{Q_1 \cup complications\ bs.\})\];

  - No drug(s) administered to the mother.

\[(E_4, Q_4) = (\{E_1 \cup blood\ sample,
blood\ glucose, NH4\}, \{Q_1 \cup complications\ bs.\})\];

- the CT scan and the blood sample, giving rise to the following problem instances.

  - Drug(s) administered to the mother.

\[(E_5, Q_5) = (\{E_1 \cup blood\ sample, measured\ peth., measured\ val.,
blood\ glucose, NH4, CT, CT\ result\},
\{Q_1 \cup complications\ bs., complications\ CT\})\].

  - No drug(s) administered to the mother.

\[(E_6, Q_6) = (\{E_1 \cup blood\ sample,
blood\ glucose, NH4, CT, CT\ result\},
\{Q_1 \cup complications\ bs., complications\ CT\})\].

The above problem instance increments are handled as discussed in Section 4.6.1. Regarding the the reduction of the number of computations involved in the various propagations though the decision networks, we restrict the discussion to mentioning that the directed acyclic graphs of the decision networks
used provide a rough indication. Detailed discussions concerning the effects of problem-directed decomposition can be found in Chapter 5.

The directed acyclic graph of the relevant decision network $D^{(E_2, Q_2)}$ is shown in Figure 6.9. We note the similarity to the directed acyclic graph shown in Figure 6.7. The difference during the evaluation is that the impact of the test result has been propagated through the decision network.

The directed acyclic graph of the relevant decision network $D^{(E_3, Q_3)}$ is shown in Figure 6.10.

The directed acyclic graph of the relevant decision network $D^{(E_5, Q_5)}$ is shown in Figure 6.11.

In cases where no drug(s) have been administered to the mother, the value of taking a blood sample is calculated in decision networks with directed acyclic graphs as shown in Figure 6.8 and Figure 6.10, where the following vertex set has been pruned.

\{peth. cons. baby, val. cons. baby, measured peth., measured val., drugs\}

### 6.6.3 Concluding remarks

The purpose of the discussion on the CNMF decision network must be considered to be illustrative of the method of problem-directed decomposition. The CNMF Bayesian network has been extended and has thus become the CNMF decision network. The computational complexity of the evaluation has been recognized to have two sources: the computational complexity inherent to probabilistic inference in general Bayesian belief networks, and the computational complexity of the full evaluation of any possible sequence of actions. We demonstrated that the method of problem-directed decomposition can be applied in order to decrease the computational effort involved in probabilistic inference in general Bayesian belief networks. To decrease the computational effort necessary in the case of the full evaluation, an additional control mechanism is needed. We envision that this control strategy will be represented separated from the decision network. The decision network, then, is activated by the control strategy to perform an heuristic evaluation. Finally, it has been demonstrated that evaluation heuristics can very well be formulated as as to successively solving a series of problem instances.
Figure 6.9: The directed acyclic graph of the relevant decision network $D^{(E_2, Q_2)}$. 
Figure 6.10: The directed acyclic graph of the relevant decision network $D^{(F_3,Q_3)}$. 
Figure 6.11: The directed acyclic graph of the relevant decision network $D^{(E_5,Q_5)}$. 
Decision networks
Chapter 7

Discussion and directions for further research

Throughout the present thesis, each chapter has been furnished with specific discussions. The purpose of the present chapter is to focus on the discussion of the main results and to indicate prospects for future research. We start with a chronological discussion of the relevant chapters.

7.1 Discussion

In Chapter 1, the Bayesian belief network formalism is sketched briefly as a framework to represent and to manipulate knowledge about a specific application domain. The knowledge is cursed with uncertainty, and manipulation of the knowledge, also called reasoning, is conducted by a (probabilistic) inference mechanism. After a reasoning session has been finished, the Bayesian belief network provides us with the updated probability of concepts discerned in the application domain. The consistency of the (updated) joint probability distribution represented by the numerical part of the Bayesian belief network has been emphasized, as well as the fact that this consistency applies to the complete model. Furthermore, the computational complexity of the probabilistic inference has been acknowledged.

Chapter 2 elaborated on the connection between dependency models and graphical representations. It was shown that the directed acyclic graph is capable of representing a wide range of (joint) probability distributions in a factorized form. In addition, the connection between the joint probability distribution, on
the one hand, and the directed acyclic graph, on the other hand, was made with the d-separation criterion.

In Chapter 3, probabilistic inference in Bayesian belief networks was discussed. We discussed Pearl's method of probabilistic inference in singly connected Bayesian belief networks, and probabilistic inference in associate representations of Bayesian belief networks with directed acyclic graphs of general topology, called junction trees. Both methods aim at utilizing the dependency structure as represented in the directed acyclic graph such as to reduce the computational complexity of probabilistic inference. The computational complexity of probabilistic inference in singly connected networks is only modest. However, in general Bayesian belief networks this is not the case. The computational complexity of probabilistic inference in the junction tree associated to some multiply connected Bayesian belief network is exponential. Further, the actual number of operations that must be performed to propagate the impact observations have on the remaining variables, largely depends on the fill-in links added by the triangulation algorithm in order to obtain a decomposable graph.

In Chapter 4, we proposed a new method to handle the computational complexity of probabilistic inference, called problem directed decomposition. The method is based on the notion of a problem instance, which consists of a set of known variables, the evidence variables, and a set of variables of interest, the query variables. The proposed method decomposes the Bayesian belief network into a network relevant to the calculation of the solution of the problem instance, that is the updated probability distribution of the query variables given the evidence variables, and the rest network that bears no impact on the solution of the current problem instance. The decomposition of the networks is performed fast, its computational complexity is proportional to the number of edges in the moral graph. Moreover, the solution of the problem instance as calculated in the relevant network is identical to the solution of the problem instance as calculated in the original network. This means that no information that has an impact on the solution of the problem instance is excluded from the calculations.

Generally, the relevant network will be multiply connected and the junction tree representation is used to enable probabilistic inference. Actually, a collection of junction trees may be associated to a relevant network, for it may very well be composed of (disconnected) components.
Concerning the triangulation algorithms used in the construction process, we mention that the decomposition into components enables the use of different heuristic triangulation algorithms with the optimization properties best suited for the discerning components.

Further, the relevant junction forest can easily be adapted after a problem instance increments, which increments consist of the addition of evidence variables, the addition of query variables, the retraction of query variables, or a combination of the three. This flexibility is a strong argument in favor of the application of the proposed decomposition method under all circumstances where the computational complexity of the probabilistic inference is an issue. The relevant junction forest can be viewed as a necessary base for the easy handling of problem instance increments, that is, without the need for rebuilding (certain junction trees of) the junction forest. Responsible for this flexibility is the inclusion of evidence-query paths in the relevant graph between weakly independent evidence and query variables.

Concerning the path inclusion mentioned above, we note that, in cases where the path \( p \) between weakly independent evidence and query variables is part of a simple loop contained in the relevant graph, its exclusion would result in a relevant junction forest such that here are problem instance increments that demand restoring to the original graph, in order to rebuild the associated junction tree. The junction tree construction, then, would include the triangulation of the newly created loop.

In the other cases, where \( p \) is not included in a simple loop in the relevant graph, exclusion of \( p \) would result in a junction forest in which problem instance increments that render \( p \) active can easily be handled. However, the test to detect whether or not \( p \) is part of a simple loop is costly. In addition, the inclusion in the relevant graph of a path \( p \) that is not included in a simple loop does not weigh heavy on the computational load, since it amounts to the addition of a singly connected subgraph to the relevant graph. For these reasons, the method of problem directed decomposition does not distinguish between the two cases.

As a consequence of the flexibility requirement, the decomposition of the original directed acyclic graph into relevant and rest subgraphs or components is such that the channel over which a component in the relevant graph may communicate with a component in the rest graph consists of precisely one joining vertex. Consequently, the best decomposition is found when the directed acyclic graph consists of vertex clusters, possibly densely connected, organized in a poly-tree
structure of interconnections. A problem instance, then, should be located in a small number of clusters, preferably in only one cluster.

In Chapter 5, the impact of problem directed decomposition for classes of randomly generated networks that differ in the connectivity density was analyzed. To this end, for each class, several experiments with the method of problem directed decomposition were performed, each with 100 networks selected from the class. In these experiments the problem instance varied. The results of the experiments confirm the behavior expected of problem directed decomposition. With increasing connectivity density of the networks, the average gain, as a result of the formation of evidence-query paths or focusing, diminishes. At the same time, the increase in the average gain as a result of evidence absorption grows. In addition, for classes with 20 vertices and a number of arcs that equals or exceeds 30, for one piece of entered evidence and one query variable, saturation has already been reached. This last result should be taken as an indication that the topological properties of the more densely connected networks with respect to loops are such that almost all vertices are included in at least one loop, which hinders the decomposition into components. The real-life Alarm network shows the topological properties mentioned above, and the effect of problem directed decomposition is accordingly. Fortunately, loops are not always that extended. The experiments with the CNMF network demonstrate the effects of problem directed decomposition in the case that the loops are locally.

Concerning the ability of randomly generated networks to provide an insight into the impact of applying the method of problem directed decomposition, we arrived at the conclusion that several hypothetical properties of the method have been verified. The results obtained from the experiments, however, cannot be exploited to make detailed predictions of the method’s behavior in real-life networks that do not incorporate a random directed acyclic graph. The alternative experimental set-up that does not involve the use of randomly generated networks, but instead, uses real-life networks was practically impossible, since at present only a few full-scale realistic networks are available.

Another possibility would be the classification of the few real-life networks such that test cases of networks to be used in extensive experimental evaluations could be generated such as to possess topological properties corresponding to the different classes. However, the real-life networks show great variety in the topology of the directed acyclic graph [71]. This variety is not only explained by the
7.2 Discussion

differences in the respective domains: restriction on the topological properties of the directed acyclic graph, in order to enable a specialized reasoning method suited for the particular application, often play a major role as well. As has already been argued, the need for modeling restrictions may evaporate as the research on reasoning methods provides us with ways to perform efficient probabilistic inference. Therefore, the setting up experimental evaluation along the lines mentioned above would run the risk of tuning the topological properties of the directed acyclic graph in the test cases to meet requirements that do not have to be met in future realistic networks.

We emphasize that the problems identified will exist in all kinds of experimental evaluations of heuristic methods using randomly generated networks, where the topology of the networks is of major importance to the results of the experiments.

Chapter 6 deals with the extension of Bayesian belief networks with decision variables and utility variables to enable the planning of action. An extended Bayesian belief network is called a decision network. It has been argued that the separation of knowledge about (causal) relations and knowledge about strategies enables the flexible planning dedicated to the specific decision problem at hand, in which new incoming information about the decision problem is taken into account immediately. Moreover, the modeling process is not obscured by the need to represent possible strategies.

The evaluation of a decision network is computationally expensive, since a combinatorial amount of propagations through the decision network is required. The computational load can be relieved by restricting the propagation the impact actions have on the remaining variables in the decision network to the utility variables only. The restriction can be achieved by applying the method of problem directed decomposition with the problem instance where the evidence variables correspond to the decision variables, and the query variables correspond to the utility variables.

An additional reduction of the number of computations involved in the evaluation of a decision network can be achieved by using heuristic evaluation algorithms such as myopic planning and subgoaling. It has been demonstrated that subgoaling can very well be formulated as the successive solving of a series of problem instances. The demonstration of problem directed decomposition in the context of decision networks nicely illustrated both its generality and its flexibility.
7.2 Direction of further research

Considering the conclusions reached in presented research, we make the following suggestions for further research.

Concerning the method of problem directed decomposition, reconsidering and adjusting the design requirements opens possibilities to develop different decomposition methods. Since we are reluctant to give in on the generality and flexibility of a decomposition method, we suggest to invest in relaxing the exactness requirement.

The exactness requirement demands that a subgraph is disconnected from the relevant graph only if the connection consists of one joining vertex. Both from the results of the experiments with the classes for randomly generated networks, and from the experiments with the Alarm network, it has become clear that the existence of loops in the directed acyclic graph may result in the inclusion of clusters that do not contain any variables from the problem instance itself.

When relaxing the exactness requirement, approximate reasoning allows a decomposition where joining connections may consist of multiple vertices, and the above-mentioned clusters may be separated from the relevant graph. As a result, an even smaller relevant graph may be expected, and, accordingly, a higher impact of the decomposition method on the computational load during probabilistic inference.

In this case, the decomposition can no longer be based only on the dependency structure of the Bayesian belief network as reflected in the directed acyclic graph. The numerical part must be taken into account to identify the strength of the joining connections with multiple vertices, so as to calculate bounds on the approximation made during probabilistic inference in the relevant junction forest.

Concerning the evaluation problem, we make the following remarks and suggestions for further investigations.

The results of the experimental evaluation of the problem directed decomposition method, have indicated the limited suitability of the use of randomly generated networks in the experimental evaluation of any method where the topology of the directed acyclic graph is predominant for the performance of the method. A possible approach to solving this problem would be the use of real-life networks in experimental evaluations. Although we are aware that this approach has its own disadvantages, we still recommend the creation of a generally accessible data
7.2 Direction of further research

base containing full-scale Bayesian belief networks.

Equally important is the construction of benchmark classes of Bayesian belief networks. Preferably, the benchmark classes should be independent of any method for which the classes will be used as the test bed. These benchmark classes should probably be set out along the lines of topological properties of the directed acyclic graph such as, for instance, the connectivity and distribution/extension of loops. Additionally, properties of the state spaces of the variables included in the networks might be of importance, as well as the type of reasoning task to be performed. The construction of these benchmark classes, however, is by no means trivial, but we are sure that the availability of benchmarks will prove to be very valuable. Moreover, we expect that research along these lines will provide insight into desirable network properties for different reasoning tasks. To the best of our knowledge, literature on this subject has not yet emerged.
Discussion and directions for further research
Bibliography


Bibliography


Appendix A

Triangulation

Most of the definitions, theorems and lemmas presented in this section originate from graph theory. For a complete introduction to graph theory, we refer to [8] and a reference on triangulated graphs is [13].

Vertex orderings

Figure A.1: Graph of the running example. The computation of the fill-in was based on the ordering shown in b. Note that in c the ordering is no longer an MCS ordering, but it yields a perfect ordering of the vertices of the graph in c.

Definition A.0.1 (Total ordering) Let $G = (V(G), E(G))$ be an undirected graph in which $V(G)$ contains $n$ vertices and $\alpha = [V_1, V_2, \ldots, V_n]$ be a total ordering of all the vertices in $V(G)$. If $V_i$ is labeled before $V_j$ the ordering dictated that
$V_i \prec V_j$.

**Definition A.0.2 (Perfect ordering)** Let $G = (V(G), E(G))$ be an undirected graph in which $V(G)$ contains $n$ vertices. An ordering $\alpha = [V_1, V_2, \ldots, V_n]$ of all the vertices in $V(G)$ is called perfect if for every $i \geq 2$:

$$\text{Adj}(V_i) \cap \{V_1, V_2, \ldots, V_{i-1}\}$$

is a complete subset of $G$.

**Example A.0.1** Consider Figure A.1.c, we have that:

- $\text{Adj}(B) = \{A, E, C\}$
- $\text{Adj}(C) = \{B, E, G, H, D\}$
- $\text{Adj}(D) = \{C\}$
- $\text{Adj}(E) = \{B, F, G, C\}$
- $\text{Adj}(F) = \{E, G\}$
- $\text{Adj}(G) = \{C, E, F, H\}$
- $\text{Adj}(H) = \{C, G\}$

Further,

- $\text{Adj}(B) \cap \{A\} = \{A\}$ is a complete subset
- $\text{Adj}(C) \cap \{A, B\} = \{B\}$ is a complete subset
- $\text{Adj}(D) \cap \{A, B, C, E\} = \{C\}$ is a complete subset
- $\text{Adj}(E) \cap \{A, B, C\} = \{B, C\}$ is a complete subset
- $\text{Adj}(F) \cap \{A, B, C, E, D, H, G\} = \{E, G\}$ is a complete subset
- $\text{Adj}(G) \cap \{A, B, C, E, D, H\} = \{C, H\}$ is a complete subset
- $\text{Adj}(H) \cap \{A, B, C, E, D\} = \{C\}$ is a complete subset

The ordering is perfect.

**Definition A.0.3 (Running intersection property)** Let $G = (V(G), E(G))$ be an undirected graph containing $P$ cliques. An ordering $[\text{Clq}_1, \text{Clq}_2, \ldots, \text{Clq}_9]$ of the cliques has the running intersection property if for every $j > 1$ there exists an $i < j$ such that

$$\text{Clq}_j \cap (\text{Clq}_1 \cup \text{Clq}_2 \cup \ldots \cup \text{Clq}_{j-1}) \subseteq \text{Clq}_i$$
Example A.0.2 Running intersection property: Consider Figure A.1.c and the clique ordering:

\[ Cl_{q_1} = \{A, B\} \]
\[ Cl_{q_2} = \{B, E, C\} \]
\[ Cl_{q_3} = \{C, D\} \]
\[ Cl_{q_4} = \{E, C, G\} \]
\[ Cl_{q_5} = \{C, G, H\} \]
\[ Cl_{q_6} = \{F, E, G\} \]

We have that,

\[ \{B, E, C\} \cap \{A, B\} = \{B\} \subseteq Cl_{q_1} \]
\[ \{C, D\} \cap \{A, B, E, C\} = \{C\} \subseteq Cl_{q_2} \]
\[ \{E, C, G\} \cap \{A, B, E, C, D\} = \{C\} \subseteq Cl_{q_2} \]
\[ \{C, G\} \subseteq Cl_{q_3} \]
\[ \{C, G, H\} \cap \{A, B, E, C, D, G\} = \{C, G\} \subseteq Cl_{q_4} \]
\[ \{F, E, G\} \cap \{A, B, E, C, D, G, H\} = \{E, G\} \subseteq Cl_{q_4} \]

This clique ordering possesses the running intersection property.

Theorem A.0.1 If the vertices of an undirected graph \( G = (V(G), E(G)) \) are numbered with a perfect ordering \( \alpha \), and the cliques are ordered according to their highest labeled vertex, then the clique ordering has the running intersection property.

Example A.0.3 Highest numbered vertex ordering: Consider Figure A.1.c and the clique ordering of Example A.0.2. The ordering is according to the highest numbered vertex of a clique, as can be seen from Table A.1.

Decomposable and triangulated graphs

Obtaining specific vertex or clique orderings as discussed above requires a specific graph class, the class of decomposable graphs.

Definition A.0.4 (Decomposition) Let \( G = (V(G), E(G)) \) be an undirected graph. A pair of subset \( X, Y \subseteq V(G) \) forms a decomposition of \( G \) if \( V(G) = X \cup Y \), and \( X \cap Y \) is a complete subset, and \( X \cap Y \) separates \( X - Y \) and \( Y - X \). A decomposition is proper if \( X \subset V(G) \) and \( Y \subset V(G) \).
Table A.1: Clique ordering according to their highest labeled vertex.

<table>
<thead>
<tr>
<th>highest labeled vertex</th>
<th>clique</th>
<th>clique number</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>{C, G, H}</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>{A, B}</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>{B, E, C}</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>{F, E, G}</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>{E, C, G}</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>{C, D}</td>
<td>3</td>
</tr>
</tbody>
</table>

**Definition A.0.5 (Decomposable graph)** Let \(G = (V(G), E(G))\) be a undirected graph. The graph \(G\) is said to be decomposable if it is complete, or if there is a proper decomposition \((A, B)\) into induced decomposable subgraphs \(G_A\) and \(G_B\).

**Example A.0.4** Decomposable graph: Consider Figure A.1.c. From Example A.0.2 we have that \((Clq_1, Clq_2, \ldots, Clq_9)\) is a decomposition sequence of \(G\).

**Lemma A.0.1** Let \(G\) be an undirected graph. \(G\) is decomposable if and only if it permits a perfect ordering of its vertices.

**Example A.0.5** Triangulated graph: Figure A.1.c is triangulated, Figure A.1.b is not triangulated. The cycle \([F, E, C, G]\) is simple and of length four and does not possess a chord.

**Theorem A.0.2** Let \(G = (V(G), E(G))\) be an undirected graph. The graph \(G\) is decomposable if and only if \(G\) is triangulated.

**Finding vertex orderings**

**Algorithm A.0.1 (Maximum Cardinality Search (MCS))** Let \(G = (V(G), E(G))\) be an undirected graph.

An ordering of the vertices in \(V(G)\) according to the MCS is obtained by starting at an arbitrary vertex and numbering this vertex 1, the next vertex to number is the one with the maximum number of previously numbered neighbors.

Proceed until all vertices have been numbered.
Example A.0.6 Maximal cardinality search: The numbering of Figure A.1.b is obtained by maximal cardinality search.

Finding a perfect vertex ordering

An MCS ordering is not necessarily perfect. In the next section we discuss an algorithm that renders an arbitrary ordering perfect by adding edges to the graph, also called filling in the graph (Algorithm A.0.2 and Theorem A.0.3).

The obtained graph, called the elimination graph (Definition A.0.6) is triangulated (Theorem A.0.4) and therefore decomposable (Theorem A.0.2). And because it is decomposable, it permits a perfect ordering of the vertices (Lemma A.0.1).

Algorithm A.0.2 (Fill-in) Let \( G = (V(G), E(G)) \) be an undirected graph with \( n \) vertices. Let \( \alpha \) be a total ordering of the vertices in \( V(G) \).

Add fill-in edges: For every vertex numbered \( i, 1 \leq i \leq n \) add edges between all neighbors of \( i \) that are assigned a lower number than \( i \) in \( \alpha \).

The set of edges added to \( G \) is called the fill-in and denoted by \( F(\alpha) \).

If \( F(\alpha) = \emptyset \), \( \alpha \) is called the zero fill-in of \( G \).

Example A.0.7 Fill-in: Consider Figure A.1.b. Vertex \( F \) has two neighbors \( E \) and \( G \) with a lower number. An edge will be filled-in between \( E \) and \( G \). Figure A.1.c shows the triangulated graph.

Definition A.0.6 (Elimination graph) Let \( G = (V(G), E(G)) \) be an undirected graph. The elimination graph \( G(\alpha) \) with respect to \( \alpha \) is defined by \( G(\alpha) = (V(G), E(G) \cup F(\alpha)) \).

Theorem A.0.3 Let \( G = (V(G), E(G)) \) be an undirected graph, \( \alpha \) an arbitrary ordering of the vertices and \( G(\alpha) \) the elimination graph of \( G \) relative to \( \alpha \). Then \( \alpha \) is a perfect ordering of the vertices in \( V(G) \) relative to \( G(\alpha) \).

Example A.0.8 Construction of an elimination graph yields a perfect ordering of the vertices: Although an MCS ordering is not an arbitrary ordering, we illustrate Theorem A.0.3 by using such ordering.

Consider Figure A.1.c and Example A.0.1: Two adjacency relations differ if we compare Figure A.1.b and c, for vertex \( E \) and vertex \( G \): \( \text{Adj}(E), \text{Adj}(G) \). The graph shown in Figure A.1.b does not permit a perfect ordering because:

\[
\text{Adj}(F) \cap \{A, B, C, E, D, H, G\} = \{E, G\} \quad \text{is not a complete subset}
\]

Clearly, the fill-in edge \((E, G)\) renders \(\{E, G\}\) a complete subset.
Lemma A.0.2 If $G = (V(G), E(G))$ is an undirected graph which permits a zero fill-in ordering of the vertices, then $G$ is triangulated.

Example A.0.9 Figure A.1.c permits a zero fill-in ordering and every simple cycle of length of more than three possesses a chord.

Theorem A.0.4 Let $G = (V(G), E(G))$ be an undirected graph and $\alpha$ a total ordering of the vertices $V(G)$. Then $G(\alpha)$ is triangulated.

Example A.0.10 Figure A.1.c permits a perfect ordering of the vertices and every cycle of length of more than three possesses a chord.
Appendix B

Accumulated data from the experiments

In the tables, we use the following abbreviations. The column denoted by $m$ contains the average gain, the column denoted by $d$ contains the average deviation and the column denoted by $s$ contains the number of samples included in the calculations. The number of pieces of evidence is denoted by $E$, and number of queries is denoted by $Q$.

For a set of $s$ networks of a given class and with a given problem instance with gains $\{g_1, g_2, \ldots, g_s\}$ we calculated,

$$m = \frac{1}{s} \sum_{i=1}^{s} g_i \quad \text{and} \quad d = \frac{1}{s} \sum_{i=1}^{s} |g_i - m|$$
Table B.1: Results for 100 randomly generated networks with 20 vertices and 19 edges.
<table>
<thead>
<tr>
<th>Q</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m</td>
<td>d</td>
<td>s</td>
<td>m</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>1.9</td>
<td>0.7</td>
<td>91</td>
<td>1.6</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>1.9</td>
<td>0.7</td>
<td>89</td>
<td>1.7</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>1.9</td>
<td>0.6</td>
<td>86</td>
<td>2.</td>
<td>0.7</td>
</tr>
<tr>
<td>4</td>
<td>2.3</td>
<td>1.</td>
<td>84</td>
<td>1.9</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>2.3</td>
<td>1.</td>
<td>82</td>
<td>2.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Average gain calculated for gain values $1 \leq \text{gain} \leq 10$

<table>
<thead>
<tr>
<th>Q</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m</td>
<td>d</td>
<td>s</td>
<td>m</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>234.1</td>
<td>136.3</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>208.6</td>
<td>142.9</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>154.3</td>
<td>140.3</td>
<td>14</td>
<td>272.</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>227.8</td>
<td>154.6</td>
<td>16</td>
<td>175.</td>
<td>143.7</td>
</tr>
<tr>
<td>5</td>
<td>249.3</td>
<td>124.4</td>
<td>18</td>
<td>23.8</td>
<td>9.7</td>
</tr>
</tbody>
</table>

Average gain calculated for gain values $10 \leq \text{gain} \leq \text{maximum gain}$

Table B.2: Results for 100 randomly generated networks with 20 vertices and 22 edges.
### Average gain calculated for gain values $1 \leq \text{gain} \leq 10$

<table>
<thead>
<tr>
<th>Q</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m</td>
<td>d</td>
<td>s</td>
<td>m</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>0.4</td>
<td>93</td>
<td>1.4</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>1.6</td>
<td>0.4</td>
<td>96</td>
<td>1.7</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>1.9</td>
<td>0.6</td>
<td>95</td>
<td>1.7</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>2.1</td>
<td>0.7</td>
<td>96</td>
<td>2.1</td>
<td>0.6</td>
</tr>
<tr>
<td>5</td>
<td>2.6</td>
<td>0.9</td>
<td>86</td>
<td>2.5</td>
<td>0.8</td>
</tr>
</tbody>
</table>

### Average gain calculated for gain values $10 \leq \text{gain} \leq \text{maximum gain}$

<table>
<thead>
<tr>
<th>Q</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m</td>
<td>d</td>
<td>s</td>
<td>m</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>240.5</td>
<td>189.3</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>350.8</td>
<td>324.4</td>
<td>4</td>
<td>550.5</td>
<td>536.2</td>
</tr>
<tr>
<td>3</td>
<td>291.3</td>
<td>179.2</td>
<td>5</td>
<td>17.0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>259.5</td>
<td>216.5</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>403.8</td>
<td>199.6</td>
<td>14</td>
<td>154.3</td>
<td>0</td>
</tr>
</tbody>
</table>

Table B3: Results for 100 randomly generated networks with 20 vertices and 25 edges.
### Average gain calculated for gain values $1 \leq \text{gain} \leq 10$

<table>
<thead>
<tr>
<th>Q</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>d</td>
<td>s</td>
<td>m</td>
<td>d</td>
<td>s</td>
</tr>
<tr>
<td>1</td>
<td>1.3</td>
<td>0.3</td>
<td>98</td>
<td>1.3</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>1.6</td>
<td>0.4</td>
<td>99</td>
<td>1.6</td>
<td>0.4</td>
</tr>
<tr>
<td>3</td>
<td>2.0</td>
<td>0.6</td>
<td>97</td>
<td>1.9</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>2.2</td>
<td>0.6</td>
<td>95</td>
<td>2.4</td>
<td>0.7</td>
</tr>
<tr>
<td>5</td>
<td>2.6</td>
<td>0.8</td>
<td>94</td>
<td>2.9</td>
<td>1.1</td>
</tr>
</tbody>
</table>

### Average gain calculated for gain values $10 \leq \text{gain} \leq \text{maximum gain}$

<table>
<thead>
<tr>
<th>Q</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>d</td>
<td>s</td>
<td>m</td>
<td>d</td>
<td>s</td>
</tr>
<tr>
<td>1</td>
<td>237.2</td>
<td>47.4</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>793.5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>676.4</td>
<td>436.4</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>748.8</td>
<td>384.2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>675.6</td>
<td>626.6</td>
<td>30.5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table B.4: Results for 100 randomly generated networks with 20 vertices and 30 edges.
### Average gain calculated for gain values $1 \leq \text{gain} \leq 10$

<table>
<thead>
<tr>
<th>Q</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m d s</td>
<td>m d s</td>
<td>m d s</td>
<td>m d s</td>
<td>m d s</td>
</tr>
<tr>
<td>1</td>
<td>1.3 0.3 98</td>
<td>1.4 0.3 100</td>
<td>1.3 0.3 100</td>
<td>1.4 0.3 100</td>
<td>1.3 0.3 100</td>
</tr>
<tr>
<td>2</td>
<td>1.9 0.5 98</td>
<td>1.9 0.5 100</td>
<td>1.6 0.4 100</td>
<td>1.8 0.5 100</td>
<td>1.8 0.5 100</td>
</tr>
<tr>
<td>3</td>
<td>2.3 0.7 99</td>
<td>2.2 0.7 100</td>
<td>2.2 0.8 100</td>
<td>2.3 0.8 100</td>
<td>2.1 0.6 100</td>
</tr>
<tr>
<td>4</td>
<td>3.1 1.0 100</td>
<td>3.1 0.9 100</td>
<td>3.1 1.1 100</td>
<td>3.1 1.0 100</td>
<td>2.9 1.1 100</td>
</tr>
<tr>
<td>5</td>
<td>3.7 1.2 95</td>
<td>3.5 1.1 99</td>
<td>3.7 1.2 99</td>
<td>4.1 1.6 100</td>
<td>3.4 1.2 99</td>
</tr>
</tbody>
</table>

### Average gain calculated for gain values $10 \leq \text{gain} \leq \text{maximum gain}$

<table>
<thead>
<tr>
<th>Q</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m d s</td>
<td>m d s</td>
<td>m d s</td>
<td>m d s</td>
<td>m d s</td>
</tr>
<tr>
<td>1</td>
<td>1274.5 1254. 2</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>5161. 2792.5 2</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>3</td>
<td>4210.5 0 1</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>4</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>5</td>
<td>1620.8 697.1 5</td>
<td>10.2 0 1</td>
<td>11.6 0 1</td>
<td>0 0 0</td>
<td>10.9 0 1</td>
</tr>
</tbody>
</table>

Table B.5: Results for 100 randomly generated networks with 20 vertices and 35 edges.
Appendix C

Additional specifications for the real-life networks CNMF and Alarm

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S9</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no</td>
<td>&lt;12mmL</td>
<td>&gt;6mmL</td>
<td>normal</td>
<td>36-42wk</td>
<td>no</td>
<td>no</td>
<td>&gt;10h</td>
</tr>
<tr>
<td>mild</td>
<td>yes</td>
<td>12-15mmL</td>
<td>4-6mmL</td>
<td>elevated</td>
<td>34-36wk</td>
<td>extrem</td>
<td>low</td>
<td>5-10h</td>
</tr>
<tr>
<td>moderate</td>
<td>15-30mmL</td>
<td>2-4mmL</td>
<td>too high</td>
<td>30-34wk</td>
<td>high</td>
<td></td>
<td>2-5h</td>
<td></td>
</tr>
<tr>
<td>severe</td>
<td>&gt;30mmL</td>
<td>&lt;1mmL</td>
<td></td>
<td>26-30wk</td>
<td>23-26wk</td>
<td></td>
<td></td>
<td>&lt;2h</td>
</tr>
</tbody>
</table>

Table C.1: The state spaces used in the CNMF model.

List of vertex names and descriptions of the Alarm model

- **anaphylaxis**: an acute or exaggerated allergic response in a sensitized host
- **anoxia**: lack of oxygen in the circulating blood or in the tissues
- **catecholamine**: drug administered to raise peripheral vascular resistance
- **hyperamonemia**: a greater than normal concentration of ammonium in plasma
- **hypoglycemia**: subnormal concentration of glucose in the blood
Table C.2: The variables contained in the CNMF model with their state spaces.

<table>
<thead>
<tr>
<th>Vertex name</th>
<th>state space</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT : S1</td>
<td></td>
</tr>
<tr>
<td>TraumaNM : S1</td>
<td>Trauma Cause : S1</td>
</tr>
<tr>
<td>hyperammonimia : S3</td>
<td>Vitamin K : S2</td>
</tr>
<tr>
<td>water load : S5</td>
<td>glyamNM : S1</td>
</tr>
<tr>
<td>immaturity : S6</td>
<td>glucose infusion : S2</td>
</tr>
<tr>
<td>drugs : S8</td>
<td>prolonged labour : S7</td>
</tr>
<tr>
<td>peth. conc. baby : S8</td>
<td>measured peth. : S8</td>
</tr>
<tr>
<td>TraumaNM-glyamNM : S1</td>
<td>time lapse (2X) : S9</td>
</tr>
<tr>
<td>NH4 : S3</td>
<td>CNMFFailure : S1</td>
</tr>
<tr>
<td></td>
<td>Trauma Manifest : S1</td>
</tr>
<tr>
<td></td>
<td>hypolytic tendency : S4</td>
</tr>
<tr>
<td></td>
<td>hypoglycemia : S4</td>
</tr>
<tr>
<td></td>
<td>liver dysfunction : S1</td>
</tr>
<tr>
<td></td>
<td>congenital disorder : S2</td>
</tr>
<tr>
<td></td>
<td>measured val. : S8</td>
</tr>
<tr>
<td></td>
<td>peth. to mother : S8</td>
</tr>
<tr>
<td></td>
<td>art. ventilation : S2</td>
</tr>
<tr>
<td></td>
<td>anoxia : S1</td>
</tr>
<tr>
<td></td>
<td>Brain Edema : S1</td>
</tr>
<tr>
<td></td>
<td>treatment : S2</td>
</tr>
<tr>
<td></td>
<td>blood glucose : S4</td>
</tr>
<tr>
<td></td>
<td>Steroid : S2</td>
</tr>
<tr>
<td></td>
<td>drugsNM : S1</td>
</tr>
<tr>
<td></td>
<td>val. conc. baby : S8</td>
</tr>
<tr>
<td></td>
<td>val. to mother : S8</td>
</tr>
<tr>
<td></td>
<td>central cyanosis : S1</td>
</tr>
</tbody>
</table>

Table C.3: The state spaces used in the Alarm model.

<table>
<thead>
<tr>
<th>The possible state spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>false</td>
</tr>
<tr>
<td>high</td>
</tr>
<tr>
<td>high</td>
</tr>
</tbody>
</table>

hypovolemia
abnormal reduction in the circulating blood volume

List of abbreviations used in the Alarm model

CO
Cardiac output
CVP
central venous pressure
LEVD
left ventricular end-diastolic volume
LV failure
left ventricular failure
MV
minute ventilation
PA SAT
pulmonary artery oxygen saturation
<table>
<thead>
<tr>
<th>Vertex name</th>
<th>state space</th>
</tr>
</thead>
<tbody>
<tr>
<td>hypovolemia : S1</td>
<td>LV failure : S1</td>
</tr>
<tr>
<td>CVP : S2</td>
<td>PCWP : S2</td>
</tr>
<tr>
<td>: S3</td>
<td>insufficient anesthesia : S1</td>
</tr>
<tr>
<td>CO : S5</td>
<td>catecholamine : S2</td>
</tr>
<tr>
<td>HR EKG : S2</td>
<td>HR SAT : S2</td>
</tr>
<tr>
<td>disconnection : S1</td>
<td>vent. machine : S3</td>
</tr>
<tr>
<td>shunt : S5</td>
<td>intubation : S4</td>
</tr>
<tr>
<td>SaCO2 : S1</td>
<td>vent. alveoli : S3</td>
</tr>
<tr>
<td>PA SAT : S2</td>
<td>min. vol. : S3</td>
</tr>
<tr>
<td>CO2 : S2</td>
<td>Exp. CO2 : S3</td>
</tr>
<tr>
<td>history : S1</td>
<td>anaphylaxia : S1</td>
</tr>
<tr>
<td>TPR : S2</td>
<td>error low output : S1</td>
</tr>
<tr>
<td>HR BP : S2</td>
<td>error cauter : S1</td>
</tr>
<tr>
<td>LVED volume : S2</td>
<td>machine vol. : S8</td>
</tr>
<tr>
<td>stroke volume : S2</td>
<td>vent. tube : S3</td>
</tr>
<tr>
<td>blood pressure : S2</td>
<td>vent. lung : S3</td>
</tr>
<tr>
<td>pulmonary embolus : S1</td>
<td>FIO2 : S5</td>
</tr>
<tr>
<td>PAP : S8</td>
<td>HR : S2</td>
</tr>
<tr>
<td>kinked tube : S1</td>
<td>pressure : S3</td>
</tr>
<tr>
<td>PAP : S8</td>
<td>Arterial : S5</td>
</tr>
</tbody>
</table>

Table C.4: The variables contained in the Alarm model with their state spaces.

- PAP: pulmonary artery pressure
- PCWP: pulmonary capillary wedge pressure
- Pressure: breathing pressure
- TPR: total peripheral resistance
Additional specifications for the real-life networks CNMF and Alarm
Appendix D

Bending in the test strategy

The notation of the updated probability, as calculated in the decision network as shown in Figure 6.3, has been augmented with a conditional part to indicate the current instantiations.

The test utility function for the test strategy is calculated as follows.

\[ F_t(t) = \]
\[ \text{MAX}[\{P(\neg cp \mid t \wedge tr \wedge pos) + P(\neg se \mid tr \wedge pos) + P(\neg dc \mid tr \wedge pos)\} \times P(pos),]
\[ \{P(\neg cp \mid t \wedge \neg tr \wedge pos) + P(\neg se \mid \neg tr \wedge pos) + P(\neg dc \mid \neg tr \wedge pos)\} \times P(pos)] + (D.1) \]
\[ \text{MAX}[\{P(\neg c \mid tr \wedge neg) + P(\neg se \mid tr \wedge neg) + P(\neg dc \mid tr \wedge neg)\} \times P(neg),]
\[ \{P(\neg c \mid \neg tr \wedge neg) + P(\neg se \mid \neg tr \wedge neg) + P(\neg dc \mid \neg tr \wedge neg)\} \times P(neg)] \]

The test strategy bands because the strategy to treat, even after a negative test result has been observed, becomes optimal.

Consider the first term of Equation D.1. Analyzing this first maximization, we can calculate for which values of \( \mathcal{P}(d) \) treatment will be postponed, even after a positive test result. Therefore we must solve the following inequality.

\[
P(\neg se \mid tr \wedge pos) + P(\neg dc \mid tr \wedge pos) < \quad (D.2)
\]
\[
P(\neg se \mid \neg tr \wedge pos) + P(\neg dc \mid \neg tr \wedge pos)
\]

Calculation of \( P(\neg dc \mid \neg tr \wedge pos) \) Using the formulas for probabilistic inference in single connected networks (Equations 3.4,3.5,3.1)we get the following equation.

\[
\lambda(R) = (1,0) \Rightarrow \lambda_R(D) = \alpha \times (\mathcal{P}(pos \mid d), \mathcal{P}(pos \mid \neg d))
\]

then

\[
\pi_{DC}(D) = \beta \times ((\mathcal{P}(d), \mathcal{P}(\neg d)) \times (\mathcal{P}(pos \mid d), \mathcal{P}(pos \mid \neg d)))
\]

175
where $\beta = \mathcal{P}(d) \ast \mathcal{P}(\text{pos} \mid d) + \mathcal{P}(\neg d) \ast \mathcal{P}(\text{pos} \mid \neg d)$. The belief function of $DC$ is calculated as follows.

$$P(\neg dc \mid \neg tr \wedge \text{pos}) = \mathcal{P}(\neg dc \mid d \wedge \neg tr) \ast \pi_{DC}(d) + \mathcal{P}(\neg dc \mid \neg d \wedge \neg tr) \ast \pi_{DC}(\neg d)$$

Substitution of $\pi_{DC}(D)$ in the previous equations yields the following.

$$P(\neg dc \mid \neg tr \wedge \text{pos}) =$$

$$\frac{\mathcal{P}(d) \ast \mathcal{P}(\text{pos} \mid d)}{\mathcal{P}(\neg d) \ast \mathcal{P}(\text{pos} \mid \neg d)} \ast \pi_{DC}(d) + \mathcal{P}(\neg dc \mid \neg d \wedge \neg tr) \ast \pi_{DC}(\neg d)$$

Let

$$A = \frac{\mathcal{P}(d) \ast \mathcal{P}(\text{pos} \mid d)}{\mathcal{P}(d) \ast \mathcal{P}(\text{pos} \mid d) + \mathcal{P}(\neg d) \ast \mathcal{P}(\text{pos} \mid \neg d)}$$

$$B = \frac{\mathcal{P}(\neg d) \ast \mathcal{P}(\text{pos} \mid \neg d)}{\mathcal{P}(d) \ast \mathcal{P}(\text{pos} \mid d) + \mathcal{P}(\neg d) \ast \mathcal{P}(\text{pos} \mid \neg d)}$$

(D.3)

After substitution of $A$ and $B$ we get the following equation.

$$P(\neg dc \mid \neg tr \wedge \text{pos}) = \mathcal{P}(\neg dc \mid d \wedge \neg tr) \ast A + \mathcal{P}(\neg dc \mid \neg d \wedge \neg tr) \ast B$$

(D.4)

**Calculation of $P(\neg dc \mid \text{pos} \wedge tr)$**

Analogous we get

$$P(\neg dc \mid tr \wedge \text{pos}) = \mathcal{P}(\neg dc \mid d \wedge tr) \ast A + \mathcal{P}(\neg dc \mid \neg d \wedge tr) \ast B$$

(D.5)

Substitution of Equations D.4 and D.5 in D.3 and substitution of the probabilities from Table 6.1 gives:

$$A < 0.13 \Rightarrow \mathcal{P}(d) < 0.003$$

This means that ignoring a positive test result is practically never allowed.

Consider the second term of Equation D.1. Analyzing this second maximization, we can calculate for which values of $\mathcal{P}(d)$ treatment is indicative after a negative test result. Therefore we must solve the following inequality.

$$P(\neg se \mid tr) + P(\neg dc \mid tr \wedge neg) > P(\neg se \mid \neg tr) + P(\neg dc \mid \neg tr \wedge neg)$$

(D.6)

**Calculation of $P(\neg dc \mid \neg tr \wedge neg)$** Using the formulas for probabilistic inference in single connected networks (Equations 3.4,3.5,3.1)we get the following equation.

$$\lambda(R) = (0,1) \Rightarrow \lambda_{R}(D) = \alpha \ast (\mathcal{P}(\text{neg} \mid d), \mathcal{P}(\text{neg} \mid \neg d))$$
then

\[ \pi_{DC}(D) = \beta * ((P(d), P(\neg d)) * (P(neg | d), P(neg | \neg d))) \]

where \( \beta = P(d) * P(neg | d) + P(\neg d) * P(neg | \neg d) \). The belief function of \( DC \) is calculated as follows.

\[ P(\neg dc | neg \land \neg tr) = P(\neg dc | d \land \neg tr) * \pi_{DC}(d) + P(\neg dc | \neg d \land \neg tr) * \pi_{DC}(\neg d) \]

Substitution of \( \pi_{DC}(D) \) in the previous equation yield the following.

\[
P(\neg dc | \neg tr \land neg) = \frac{P(d)*P(neg|d)}{P(d)*P(neg|d) + P(\neg d)*P(neg|\neg d) + P(\neg d)*P(neg|\neg d) + P(d)*P(neg|d)} \\
= \frac{P(d)*P(neg|d)}{P(d)*P(neg|d) + P(\neg d)*P(neg|\neg d) + P(\neg d)*P(neg|\neg d) + P(d)*P(neg|d)}
\]

Let

\[ C = \frac{P(d)*P(neg|d)}{P(d)*P(neg|d) + P(\neg d)*P(neg|\neg d) + P(\neg d)*P(neg|\neg d) + P(d)*P(neg|d)} \]

\[ D = \frac{P(\neg d)*P(neg|\neg d)}{P(d)*P(neg|d) + P(\neg d)*P(neg|\neg d) + P(\neg d)*P(neg|\neg d) + P(d)*P(neg|d)} \]

After substitution of \( C \) and \( D \) we get the following equation.

\[ P(\neg dc | neg \land \neg tr) = P(\neg dc | d \land \neg tr) * C + P(\neg dc | \neg d \land \neg tr) * D \quad (D.7) \]

**Calculation of \( P(\neg dc | neg \land tr) \)**

Analogous we get

\[ P(\neg dc | neg \land tr) = P(\neg dc | d \land tr) * C + P(\neg dc | \neg d \land tr) * D \quad (D.8) \]

Substitution of Equations D.7 and D.8 in D.6 and substitution of the probabilities from Table 6.1 gives:

\[ C > 0.131 \Rightarrow P(d) > 0.594 \]

Thus for an clinical likelihood of \( P(d) > 0.594 \), treatment should be initiated even after a negative test result has been observed.

Next, we prove that the bending does not influence the optimal strategy.

The utility function of the test strategy after the bending is calculated as follows.

\[
F_t(t) = \{ P(\neg cp | t) + P(\neg se | tr) + P(\neg dc | tr \land pos) \} * P(pos) + \\
\{ P(\neg cp | t) + P(\neg se | tr) + P(\neg dc | tr \land neg) \} * P(neg) \quad (D.9)
\]
We can write this equation as follows.

\[
F_t(t) = P(pos) * \{ P(-dc \mid d \land tr) * (A - C) + P(-dc \mid -d \land tr) * (B - D) \} + P(-cp \mid t) + P(-se \mid tr) + P(-dc \mid d \land tr) * C + P(-dc \mid -d \land tr) * D
\]  

(D.10)

The utility function of the treat strategy is as follows.

\[
F_{tr}(tr) = P(-cp \mid -t) + P(-se \mid tr) + P(-dc \mid d \land tr) * P(d) + P(-dc \mid -d \land tr) * P(-d)
\]

We prove that bending takes place only for values of \( P(d) \) where:

\[
F_{tr}(tr) > F_t(t)
\]

(D.11)

Proof:

Equation D.11 can be rewritten as follows.

\[
P(-dc \mid d \land tr) * (C * (1 - P(pos)) + A * P(pos) - P(d)) + P(-dc \mid -d \land tr) * (D * (1 - P(pos)) + B * P(pos) - P(-d)) + P(-cp \mid t) - P(-cp \mid -t) < 0
\]  

(D.12)

where

\[
C * (1 - P(pos)) = \frac{P(d) * P(neg \mid d)}{P(d)+P(neg \mid d) + P(-d)+P(neg \mid -d)} * P(neg)
\]

\[
= \frac{P(d) * P(neg \mid d)}{P(neg)} * P(neg)
\]

\[
= P(d) * P(neg \mid d)
\]

(D.13)

and

\[
A * P(pos) = P(d) * P(pos \mid d)
\]

\[
D * (1 - P(pos)) = P(-d) * P(neg \mid -d)
\]

\[
B * P(pos) = P(-d) * P(pos \mid -d)
\]

(D.14)

Substitution of Equation D.13 and Equation D.14 in Equation D.12 yields the following inequality.
\( \mathcal{P}(\neg dc \mid d \land tr) \ast (\mathcal{P}(d) \ast \mathcal{P}(\neg d) + \mathcal{P}(\neg d) \ast \mathcal{P}(d) - \mathcal{P}(d)) + \\
\mathcal{P}(\neg dc \mid \neg d \land tr) \ast (\mathcal{P}(\neg d) \ast \mathcal{P}(\neg d) + \mathcal{P}(\neg d) \ast \mathcal{P}(\neg d) - \mathcal{P}(\neg d)) + \\
\mathcal{P}(\neg cp \mid t) - \mathcal{P}(\neg cp \mid \neg t) < 0 \\
\iff \\
\mathcal{P}(\neg dc \mid d \land tr) \ast (\mathcal{P}(d) \ast (\mathcal{P}(\neg d) + \mathcal{P}(\neg d) + \mathcal{P}(\neg d) + \mathcal{P}(\neg d) - 1)) + \\
\mathcal{P}(\neg dc \mid \neg d \land tr) \ast (\mathcal{P}(\neg d) \ast (\mathcal{P}(\neg d) + \mathcal{P}(\neg d) + \mathcal{P}(\neg d) + \mathcal{P}(\neg d) - 1)) + \\
\mathcal{P}(\neg cp \mid t) - \mathcal{P}(\neg cp \mid \neg t) < 0 \\
\iff \\
\mathcal{P}(\neg cp \mid t) - \mathcal{P}(\neg cp \mid \neg t) < 0 \\
\]

It is reasonable to assume that test complications cannot occur spontaneously, therefore this condition is true. \(\square\)
Bending in the test strategy
Vraagstelling-gestuurde decompositie van Bayesian belief networks

Samenvatting

Onderzoek binnen de Kunsmatige Intelligentie heeft inmiddels de noodzaak aangetoond van methoden voor het omgaan met onzekerheid in kennisystemen. Dit proefschrift beschouwt het gebruik van Bayesian belief networks als één van de mogelijkheden om aan bovengenoemde eis tegemoet te komen.

Een Bayesian belief network bestaat uit een gekwantificeerde gerichte acyclische graaf. De knopen corresponderen met (stochastische) variabelen, die op hun beurt concepten uit het kennis domein representeren. De takken representeren relaties tussen de variabelen zodat wederzijdse beïnvloeding mogelijk is. Door de afwezigheid van takken worden de onafhankelijkheidsrelaties gereduceerd die gelden voor de variabelen binnen het domein.

Onzekerheid wordt gereduceerd in de vorm van conditionele kansenverdelingen die de sterken van de beïnvloedingen kwantificeren, en a priori kansenverdelingen die initiële kansen van hypothesen kwantificeren. De propagatie van kansen in een Bayesian belief network is verder consistente met waarschijnlijkheidsrekening. De rekencomplexiteit van de propagatie van kansen in een Bayesian belief network is inherent exponentieel. Meer specifiek is bewezen dat probabilistische inferentie in Bayesian belief networks een NP-hard probleem is.

In dit proefschrift wordt een nieuwe decompositiemethode voor Bayesian belief networks voorgesteld. Deze decompositiemethode heeft tot doel de rekencomplexiteit beheersbaar te houden. De decompositiemethode gaat uit van een gegeven vraagstelling. Een vraagstelling bestaat uit twee verzamelingen variabelen: de aanwijzingen waarvan de waarden bekend verondersteld worden en de vragen.
We zijn geïnteresseerd in de verzameling van individuele kansverdelingen van de vragen gegeven de aanwijzingen. Deze verzameling wordt de *oplossing* van een vraagstelling genoemd.

De voorgestelde decompositiemethode deelt een Bayesian belief network op in een *relevant* netwerk en een *rest* netwerk. Het relevante netwerk is van direct belang voor het bepalen van de oplossing van de gegeven vraagstelling en het rest netwerk niet. Eenvoudig gezegd bestaat het relevante netwerk uit alle paden die in het oorspronkelijke Bayesian belief network een vraag met een aanwijzing verbinden. Het aantal rekenoperaties dat nodig is voor het bepalen van de oplossing van de vraagstelling kan beperkt blijven doordat de propagatie van de invloed die aanwijzingen hebben op de vragen alleen uitgevoerd hoeft te worden in het relevante netwerk.

In de literatuur zijn verschillende decompositiemethoden bekend. Deze methoden zijn echter niet algemeen toepasbaar en kunnen niet adequaat omgaan met wijzigingen in de vraagstelling. Bij de ontwikkeling van de nieuwe decompositiemethode zijn algemene toepasbaarheid, flexibiliteit en exacte probabilistische inferentie als ontwerpeisen gesteld.

Vanwege de eis van algemene toepasbaarheid is gekozen voor de propagatie van kansen in een *junction tree*. De junction tree inferentiemethode kan in principe toegepast worden voor de propagatie van kansen in elk meervoudig verbonden Bayesian belief network.

De eis van flexibiliteit houdt in dat de bestaande decompositie eenvoudig aangepast kan worden wanneer er wijzigingen optreden in de vraagstelling. Feitelijk betekent dit dat een (her)beschouwing van het oorspronkelijke Bayesian belief network en reconstructie van de junction tree niet nodig zijn. De junction tree van het relevante netwerk kan bijvoorbeeld (tijdelijk) worden uitgebreid met delen van de junction tree van het rest netwerk.

De eis van exacte probabilistische inferentie houdt in dat de oplossing voor een gegeven vraagstelling op basis van het relevante netwerk identiek dient te zijn aan de oplossing zoals die bepaald zou zijn in het oorspronkelijke Bayesian belief network.

De het effect van de toepassing van de voorgestelde decompositiemethode is geëvalueerd. Het *best-case* rekencomplexiteit en de *worst-case* rekencomplexiteit van de voorgestelde decompositiemethode kunnen bepaald worden aan de hand van een theoretische analyse. Voor het *gemiddelde gedrag* is dit niet het geval.
Gebaseerd op een globaal idee over topologische eigenschappen van Bayesian belief networks zijn een aantal hypothesen opgesteld over het gemiddelde gedrag van de voorgestelde decompositiemethode. Vervolgens is een experimentele evaluatie uitgevoerd teneinde de hypothesen te verifiëren.

Omdat over een indeling van de weinige bestaande Bayesian belief networks op grond van hun topologische eigenschappen in klassen vooralsnog geen consensus is bereikt is gekozen voor het experimenteren met klassen van random gegenereerde Bayesian belief networks met bijbehorende random gegenereerde vraagstellingen.

De hypothesen worden gestaafd door de resultaten uit de experimenten. Opval lend is de grote invloed van de dichtheid van de verbondenheid van de beschouwde Bayesian belief networks. Dit wordt verklaard door de aard van de random gegenereerde Bayesian belief networks.

Realistische Bayesian belief networks kunnen in elk van de beschouwde klasse bevatten zodat geen algemeen geldende conclusies getrokken kunnen worden over het gedrag van de decompositiemethode in de praktijk. Dit probleem is aanwezig bij de experimentele evaluatie van elke methode waarvan het gedrag sterk afhangt van de topologische eigenschappen van de Bayesian belief networks. Niettemin zal het gebruik van random gegenereerde Bayesian belief networks voorlopig de enige manier zijn om een experimentele evaluatie van het gemiddelde gedrag mogelijk te maken.

Het positieve effect van het toepassen van de voorgestelde decompositiemethode kan tevens aangetoond worden in bestaande Bayesian belief networks. Omdat bij de ontwikkeling van Bayesian belief networks vaak getracht wordt de rekencomplexiteit beperkt te houden door simpele topologieën af te dwingen moet opgemerkt worden dat bestaande Bayesian belief networks niet zondermeer geschikt zijn om als test netwerken te fungeren.

Experimenten zijn uitgevoerd met twee bestaande Bayesian belief networks; het Central Neural Muscular Failure (CNMF) netwerk en het Alarm netwerk. Deze netwerken bevatten elk ongeveer evenveel knopen en tijdens hun ontwikkeling zijn geen vereenvoudigingen gemaakt ten behoeve van de inferentie snelheid.

Op de schaal van best-case tot worst-case rekencomplexiteit bevindt de toepassing van de decompositiemethode in het CNMF netwerk zich in de buurt van de best-case terwijl de toepassing van de decompositiemethode in het Alarm netwerk zich in de buurt van de worst-case bevindt. Dit significante verschil wordt grotendeels verklaard door de topologische eigenschappen van de respectievelijke
netwerken. De complexiteit van het CNMF netwerk is laag ten opzichte van de complexiteit van het Alarm netwerk omdat er veel minder knopen bevatten in een of meer *lussen*. Het CNMF netwerk is bijna een boom; de twee niet overlappende lussen bevatten ongeveer een kwart van het totaal aantal knopen en zijn op te vatten als lokale verdichtingen in de topologie (clusters). In de decomposities kunnen deze twee clusters worden gescheiden waardoor het relevante netwerk zowel minder complex als minder omvangrijk is dan het oorspronkelijke Bayesian belief network. Het Alarm netwerk bevat zes lussen waarin ongeveer de helft van het totaal aantal knopen bevatt is. Deze zes lussen zijn niet op te vatten als enkele clusters, maar domineren de hele topologie; ze vormen zelf een cluster. In elke decompositie van het Alarm netwerk bevat het relevant netwerk in ieder geval dit cluster waardoor zijn complexiteit gelijk is aan de complexiteit van het oorspronkelijke Bayesian belief network en de omvang is in ieder geval de helft van het Bayesian belief network.

De toepassing van de voorgestelde decompositiemethode werd ook bestudeerd in op Bayesian belief networks gebaseerde adviesystemen. Het doel van deze adviesystemen is het suggereren van *acities*. Het proces om tot een suggestie voor een actie te komen wordt het *plannen* van acties genoemd. Het Bayesian belief network formalisme moet uitgebreid worden om het plannen van acties mogelijk te maken.

In de literatuur zijn verschillende uitbreidingen van het Bayesian belief network formalisme beschreven. In dit proefschrift wordt de uitbreiding genaamd *beslisnetwerken* beschouwd. In een beslisnetwerk wordt informatie verwerkt zodra ze beschikbaar komt, waardoor het plannen van acties zoveel mogelijk toegesplitst blijft op het onderhavige beslisprobleem.

Een beslisnetwerk is een Bayesian belief network waarbij de knopenverzameling uitgebreid is met *beslisknopen* en *utiliteitknopen*. Een beslisknoop representeert te nemen acties. Een utiliteitknoop representeert gevolgen van genomen acties. Deze gevolgen worden gewogen door utiliteiten. Tijdens het plannen wordt voor elke mogelijke actie de afweging gemaakt tussen de (verwachtte) negatieve gevolgen en de (verwachte) positieve gevolgen. Deze afweging wordt uitgedrukt in de utiliteitsmaat. Het beslisnetwerk suggereert de actie waarvan de waarde van de bijbehorende utiliteitsmaat het grootst is. Voor de berekening van de utiliteitsmaat behorend bij een actie zijn de marginale kansverdelingen nodig van de variabelen die corresponderen met de utiliteitknopen. Deze kansverdelingen
worden berekend door de actie in het beslisnetwerk in te voeren en vervolgens zijn effect te propageren door het beslisnetwerk. In het algemeen zullen tijdens deze propagatie meer rekenoperaties uitgevoerd worden dan nodig zijn voor het bepalen van de benodigde kansverdelingen.

Aangetoond wordt dat de voorgestelde decompositiemethode zonder meer toegepastbaar is in de context van beslisnetwerken. Verder is een vergelijkende studie uitgevoerd ten einde het beslisnetwerk formalisme te plaatsen in de taxonomie van methoden die het plannen van acties tot doel hebben.

Concluderend wordt gesteld dat beslisnetwerken zeer geschikt zijn om toegepast te worden in domeinen waarin het afstemmen van acties op voortdurend veranderende informatie essentieel is. Verder kan de rekenintensiviteit die nodig is voor deze flexibiliteit beheersbaar gehouden worden door de toepassing van de voorgestelde decompositiemethode.
Acknowledgements

During the years I was working towards my Ph.D., many people were prepared to help and encourage me. On the completion of this thesis, it is a pleasure to mention those people who were of particular importance during this period, and to express my gratitude for their support.

At the time I started my research, I also started as a parent. I thank Anouk for being my daughter, the fascinating spectacle of her development from baby to school-girl provided a pleasant and compelling way to take my mind off my work. I thank Jelle for sharing with me the demanding task of parenthood. At the times it seemed impossible to stand the struggle of combining work and family, he always succeeded in convincing me that pulling out would not be the solution. I also thank Han and Hetty, my parents, and Marjet, my "little" sister for their own personal way of giving me support.

It goes without saying that I am very grateful to Eric Backer who offered me the opportunity to carry out my research in his group. He provided an atmosphere of scientific freedom and never hesitated to support me in my decisions to (temporarily) spend more time on caring for Anouk at the expense of my research progress. Further I am indebted to Jan van der Lubbe for his willingness to act as my copromotor.

Although my research area is not familiar to electrical engineering students, some were prepared to carry out their graduation projects in this area. I very much enjoyed the discussions with Eric Simonse, who enthusiastically explored the topic of assisting the construction of Bayesian belief networks by learning their structure from data. In addition, I would like to thank Ramon Huls for the work he did on the implementations and experimentations as reported in Chapter 5.

I express my gratitude to Steen Andreassen for inviting me to visit Aalborg University Center. With him and Per Thorgård I had many lively discussions about the construction of a Bayesian belief network for a complex medical domain. The two exhausting visits to Aalborg were very stimulating for my research. Steen
also put me on the track of decision networks, the topic of Chapter 6. I would like to thank Per not only for his contributions as an expert in the domain of anaesthesia, but also for his hospitality in inviting me to his home, for taking me for a walk in the beautiful surrounding areas of Aalborg, and for our discussions about training dogs and horses. In Aalborg, I first met Finn Jensen. I would like to thank him for the valuable discussions we had about my decomposition method, as we met at several different locations in Europe.

Further, I am obliged to Linda van der Gaag for her critical reading of the draft thesis. This thesis certainly benefited from her comments and suggestions. I thank Bob Goedhart for his efforts at reading and commenting on the various (premature) versions of the draft thesis. In addition, indirect support came from involving me in the process of finishing his own thesis.

Finally, I am indebted to Mrs. J.B. Zaat-Jones for correcting the English text.
Curriculum Vitae

Erica van de Stadt was born in Kaduna, Nigeria, on March 13, 1962. In June 1982, she obtained her VWO diploma from the Ichtus College in Enschede.

She continued her activities with an equestrian education. In December 1982 she obtained her diploma as horseback-riding instructor from the Stichting Nederlandse Hippische Opleidingen.

She took her M.Sc. in Computer Science from the State University of Leiden, in July 1988. Her Master's project was carried out at the Philips Research Laboratories in Eindhoven and her thesis was entitled: "Contour Fitting with Linear Functions".

In January 1989, she continued her studies at the Information Theory Group of the Department of Electrical Engineering at the Delft University of Technology. In November 1991, she received her Chartered Designer's certificate in Information Technology and her thesis was entitled: "Bayesian belief networks – propagation of uncertainty". She stayed with the same group to do research towards a Ph.D. in the area of Bayesian belief networks. The emphasis was on bringing under control the number of computations involved in the propagation of uncertainty. Apart from her work, she is interested in bicycling, horses, and, last but not least, spending time with her family.