Sensitivity Analysis in Bayesian Networks and Influence Diagrams

by

Michael Höhle and
Brian Kristiansen

Group: DAT5 C1-202a

December 1998

INSTITUTE OF ELECTRONIC SYSTEMS
Department of Computer Science, Aalborg University
Fredrik Bajers Vej 7E • DK 9220 Aalborg Ø • Denmark

AALBORG UNIVERSITY
Institute of Electronic Systems
Department of Computer Science

Fredrik Bajers Vej 7, DK-9220 Aalborg Øst

Title:
Sensitivity Analysis in Bayesian Networks and Influence Diagrams

Project Period:
1. September - 21. December 1998

Semester:
Dat5 (9th semester)

Project Group:
C1-202a

Members:
Michael Höhle
Brian Kristiansen

Supervisor:
Professor Finn V. Jensen

Number of Copies: 7

Number of Pages: 77

ABSTRACT:

This report deals with performing sensitivity analysis in Bayesian networks and influence diagrams. The work of Coupé and van der Gaag, which covers analytical sensitivity analysis for a subset of Bayesian networks and sensitivity setups, is extended. Our extension permits an analytical or numerical sensitivity analysis on an arbitrary Bayesian network and any consistent sensitivity setup, i.e., non-binary variables and multiple conditional probabilities under study are allowed. We develop a theory for analytical sensitivity analysis in influence diagrams having one or two decision and multiple conditional probabilities under study. The analysis is performed by calculating a radius of change to the closest change in decision alternative. Furthermore, we have developed a tool to perform the sensitivity analysis in practice on Bayesian networks and influence diagrams constructed in Hugin. For influence diagrams the tool performs the analysis analytically in the above cases, otherwise, it uses a numerical approach that can handle any influence diagram and any consistent sensitivity setup. The tool and with that the theory is applied to an influence diagram of a fictitious milk-testing problem.

Preface

This report describes the project work performed during the fall of 1998 by project group C1-202a, 9th semester (DAT5) at the Department of Computer Science at Aalborg University. The topic of the project is to develop and apply methods — based on the general framework of decision support systems — enabling an in-depth sensitivity analysis in Bayesian networks and influence diagrams.

The report is mainly addressing the supervisor, the censor, and others with interest in the subject of sensitivity analysis in Bayesian networks and influence diagrams. It should be seen as a midway report reflecting the results obtained this semester and formulating areas requiring further research, once the work is continued.

Citations are on the form [⟨author⟩, ⟨year⟩, ⟨page reference⟩], which refers to the bibliography on page 75. Note, the ⟨page reference⟩ field is optional and can instead contain a reference to, e.g., a theorem. Figures, tables, equations, theorems and lemmas are numbered as ⟨chapter⟩.⟨number⟩, where ⟨chapter⟩ corresponds to chapter in which it appear, and ⟨number⟩ corresponds the number of appearance within the chapter, and each category having its own counter.

Notation with respect to Bayesian networks and influence diagrams and general graph theoretic notation is shortly described in Appendix A.

We would like to thank Hugin Expert A/S for providing us with a full version of Hugin during the project period.

Michael Höhle   Brian Kristiansen

Contents

Chapter 1

Introduction

The philosophy of this decade seems to be; if a human cannot perform a decision by pure reasoning we better exploit the magic power of computers. This is done by constructing computer systems to support or at the extreme to replace the human. Our trade as knowledge engineers is the design of expert systems, but unfortunately our systems are often interpreted as oracles by the users. The conclusion of an expert systems communicated by, e.g., a printout is regarded more credible than an expert's opinion. As engineers we believe in a mechanistic world view convincing us that modeling reality is possible, thus, the user's naive application of our system demands an extra effort to make the system as consistent with reality as possible. To optimize the consistency, constructing a belief model becomes an iterative process of constructing, analyzing and adjusting the model. But currently, none of of the integrated belief network construction products, such as e.g. Hugin, reflect this iterative process. The products only aid the knowledge engineer in the construction and adjusting phases. No support is given during the analyze phase! This report presents our ideas on how to equip the knowledge engineer with an aid to formalize the analyzing phase for Bayesian networks and influence diagrams.

Eliciting a probability model such as an influence diagram usually happens in co-operation between a knowledge expert and an expert. Surveys show that creating the qualitative structure of the network usually is relatively straightforward, [Coupé and van der Gaag, 1998b]. Assessing the conditional probabilities and utility values used in the quantitative part, on the other hand, is a difficult task. This is mostly due to the large number of values that have to be specified, [Coupé and van der Gaag, 1998b]. Methods for extracting these values vary from querying databases, recording previous observations to consulting experts either from literature, or simply by asking a co-operating mechanic or doctor. Inevitable these values become inaccurate, and especially the values achieved by asking a human expert are known to be quite incorrect, [Coupé and van der Gaag, 1998b]. By comparing to previously observed situations or using the experts judgment the validity of the decision of an influence diagram can be evaluated. But, as already mentioned, the many inaccurate values seldomly produce the desired results. To qualify the repair phase the knowledge engineer needs to know how entered values effect the conclusions of the model. Once this knowledge has been acquired the values can be adjusted to produce more valid results. This investigation comes under the heading of *sensitivity analysis*.

Sensitivity analysis of mathematical models in general is defined as a systematic investigation of the effect the uncertain parameters have on the output, [Coupé and van der Gaag, 1998b]. Seen from a mathematical point of view this is done by describing the output as a function of all the uncertain parameters such that these can be varied simultaneously. For Bayesian networks and influence diagrams the desired output corresponds to possible hypothesis variables or decision variables in the network. Systematically varying all the questionable values would be an computational intractable task. Instead, it is common to only consider one probability or decision of interest and investigate how it depends on a small set of parameters, typically one or two. With this reduced setup it is in Bayesian networks possible to functionally describe the exact effects of the study parameters on the chosen target probability, [Coupé and van der Gaag, 1998b], [Coupé and van der Gaag, 1998a], and [Laskey, 1993]. But so far, no theory has been developed on how to perform a sensitivity analysis in influence diagrams.

Motivated by this shortcoming, we define the concept of the *radius of change* for a decision. It indicates how much the parameters under study have to be varied from their original values, before a different decision alternative would be chosen. The goal is to be able to calculate this radius in arbitrary influence diagrams.

The present report is organized as follows. In Chapter 2 we extend the work of [Coupé and van der Gaag, 1998b] for Bayesian networks. The following two chapters investigate how sensitivity analysis can be performed in influence diagrams. The radius of change is introduced in Chapter 3 and we show how to calculate it for influence diagram with one decision and one or two parameters under study. Chapter 4 extends these calculations to influence diagrams with two decisions. Implementational details of our sensitivity analysis tool (sat) are presented in Chapter 5.

Finally, we demonstrate in Chapter 6 the applicability of the radius of change by performing a model-analyze phase. This argumentation is performed by assuming the role of a knowledge engineer using sat to undertake a sensitivity analysis in a milk testing problem. Appendix A contains general graph theoretic definitions used throughout the report and a short introduction to Bayesian networks and influence diagrams, for readers unfamiliar with the concepts of a graphical model.

Chapter 2

Sensitivity Analysis In Bayesian Networks

In this chapter we will discuss sensitivity analysis in Bayesian networks starting with the basic case of having binary variables and only one conditional probability under study [Coupé and van der Gaag, 1998b]. Hereafter, we extend this basic case to both non-binary variables and multiple variables under study. Finally, we discuss how to calculate the coefficients by using a number of propagations, as discussed in [Coupé and van der Gaag, 1998b].

2.1 Basic Sensitivity Analysis

In the basic sensitivity analysis we only consider Bayesian networks with binary variables, i.e. variables with two states and only one conditional probability under study. Let V_t be the target node and $\nu_t \in \nu(V_t)$ be the target state, furthermore, denote by V_s the node in which the conditional probability $x = P(V_s = \nu_s | \operatorname{pa}(V_s) = \pi')$ under study is located. Let O be the set of observed nodes in the network, and $o \in \Omega_O$ the observed configuration. We are now interested in how $P(v_t|o)$ behaves as a function of the conditional probability under study. If ν_t and o are in conflict it is clear that $P(v_t|o)(x) = 0$, which is not interesting. Thus, we from now on assume that the two configurations are not conflicting.

For some set, W, of nodes in the graph, simply investigating d-separation properties in the DAG, G, of the Bayesian network, can give us an idea of $P(v_t|o)(x)$ if $V_s \in W$. For this purpose we define the **sensitivity set**

$$Sen(V_t, O) = \{V_i \in V(G) \mid \neg\langle\{X_i\}|O|\{V_t\}\rangle^d_{G*}\},$$

where G^* corresponds to G except that the new node X_i is added as a parent of V_i. Now by [Coupé and van der Gaag, 1998b, Proposition 3.13], if $V \notin Sen(V_t, O)$ then $P(v_t|o)(x) = c$. Hence, it is only worth studying nodes in $Sen(V_t, O)$. For these nodes [Coupé and van der Gaag, 1998b, Proposition 4.1] states the following theorem.

Theorem 2.1

Let $B = (G, P)$ be a Bayesian network. Let $O \subseteq V(G)$ be the set of observed nodes in G and let o denote the corresponding observations. Let V_t be the target node and let $Sen(V_t, O)$ be the sensitivity set for V_t given O. Then, for any value ν_t of V_t, we have that

$$P(\nu_t|o) = \frac{ax + b}{cx + d}$$

for every conditional probability $x = P(\nu_s|\pi')$ of every node $V_s \in Sen(V_t, O)$, where a, b, c, and d are constants related to the values ν_s of V_s and π' of $\text{pa}(V_s)$.

From the theorem we can conclude that when the variables are binary, the target probability, $P(\nu_t|o)$, is a fraction of two linear functions in the conditional probability under study, x. A special case of the above theorem occurs when $(V_s \cup \text{de}(V_s)) \cap O = \emptyset$. Then, [Coupé and van der Gaag, 1998b, Proposition 4.3] states that $P(\nu_t|o)(x) = ax + b$.

2.2 Sensitivity Analysis For Non-Binary Variable

In this section we extend the Bayesian networks to have non-binary variables, i.e. more than two states per variable. First, we define a concept of how to update the probabilities, then we prove a lemma stating that this concept is a sound way to update the conditional probabilities of complementary states when ranging a conditional probability under study. Hereafter, we present the main result of this section, stating that if the probabilities in a non-binary variable are updated using this concept, $P(v_t|o)$ is again given by a fraction of two linear functions. Finally, we look at what kind of information the junction tree can provide.

Given $x = P(\nu^k|\pi')$, the normalization of complementary states in a node denotes that the ratio between any two conditional probabilities, $P(\nu^i|\pi')$ and $P(\nu^j|\pi')$ with $i, j \neq k$, is preserved as $P(\nu^k|\pi')$ is varied between zero and one. Special attention has to be given if $\exists \nu^l \neq \nu^k : P(\nu^l|\pi') = 0$ or if $x = 1$. We now give an expression for how to normalize, and then prove that it actually preserves the ratios.

Definition 2.2 (Normalization of the complements)
Assume $x = P(\nu^k|\pi') \neq 1$ and let $x^* = x + \Delta x$ be the value of x changed by Δx such that $0 \leq x^* \leq 1$. Then, for $z_i = P(\nu^i|\pi')$, $i \neq k$, the value after normalization of the complements is given by

$$z_i^* \quad = \quad \frac{z_i(1 - x^*)}{1 - x}.$$

If $x = 1$ it is not possible to perform a normalization of the complements, thus, some other notion of updating the probabilities is necessary. We abstain from covering this situation. The following lemma proves that the above update concept is sound as it actually preserves the desired ratios.

Lemma 2.3
Let $B = (G, P)$ be a Bayesian network and let $V \in V(G)$ be a node with $|\nu(V)| = m + 1$. Assume, $x = P(\nu^{m+1}|\pi')$ for $\text{pa}(V) = \pi'$, and $z_i = P(\nu^i|\pi'), i = 1, \ldots, m$. In other words, $P(V|\pi') = (z_1, \ldots, z_m, x)$. Furthermore, let $k_{ji} = \frac{z_j}{z_i}$, where $i, j \in \{1, \ldots, m\}$ be the ratios between the complements. If $z_i = 0$, k_{ji} is defined to be 0.

If the value x^* of $P(\nu^{m+1}|\pi')$ is ranged within $[0,1[$, then normalization of the complements will preserve the ratios, k_{ji}.

Proof

The idea of the proof is to show that

$$x^* + \sum_{i=1,\ldots,m} z_i^* \;=\; 1 \tag{2.1}$$

and $\forall i,j \in \{1,\ldots,m\}$ that

$$\frac{z_j^*}{z_i^*} \;=\; k_{ji}. \tag{2.2}$$

Since $\sum_{i=1,\ldots,m} z_i = 1-x$, we have that

$$x^* + \sum_{i=1,\ldots,m} z_i^* \;=\; x^* + \sum_{i=1,\ldots,m} \frac{z_i(1-x^*)}{1-x} \;=\; x^* + \frac{1-x^*}{1-x} \sum_{i=1,\ldots,m} z_i \;=\; 1$$

Thus the normalization respects (2.1).

Furthermore, we show that the normalization respects (2.2). By definition the ratios of the complements are preserved,

$$\frac{z_j^*}{z_i^*} \;=\; \frac{z_j(1-x^*)(1-x)}{z_i(1-x^*)(1-x)} \;=\; \frac{z_j}{z_i} \;=\; k_{ji}.$$

This completes the proof.

□

Before we show the main result of this section, we need a general result about how sum-marginalization of linear functions preserves the linearity.

Lemma 2.4

Let $f(x,U) = g(x,V)\cdot\prod_i g_i(U_i)$ be a function over a universe of variables $U \cup \{x\}$, where $U_i \subseteq U$ and $V \subseteq U$. Assume about g that $g(x,V) = h_1(V) + x \cdot h_2(V)$. Then, for a set of variables W, such that $W \cap \{x\} = \emptyset$, we have that

$$\sum_W f(x,U) = a(U\backslash W) + x\; b(U\backslash W). \tag{2.3}$$

Proof

By the definition of $f(x,U)$ we rewrite

$$\begin{aligned}
\sum_W f(x,U) &= \sum_W \left(h_1(V)\cdot\prod_i g_i(U_i) + x\cdot h_2(V)\cdot\prod_i g_i(U_i)\right)\\
&= \left(\sum_W h_1(V)\cdot\prod_i g_i(U_i)\right) + \left(\sum_W x\cdot h_2(V)\cdot\prod_i g_i(U_i)\right)\\
&= \left(\sum_W h_1(V)\cdot\prod_i g_i(U_i)\right) + x\left(\sum_W h_2(V)\cdot\prod_i g_i(U_i)\right)
\end{aligned}$$

Now, let $a(U \backslash W) = \sum_W h_1(V) \prod_i g_i(U_i)$ and $b(U \backslash W) = \sum_W h_2(V) \prod_i g_i(U_i)$. This concludes the proof.

□

In the case of non-binary variables, we can now prove the theorem stating that $P(\nu_t|o)$ is a fraction of two linear-functions in x. The proof of this theorem is an extension of the proof of Theorem 2.1.

Theorem 2.5

Let $B = (G, P)$ be a Bayesian network and $O \subseteq V(G)$ the set of observed nodes with the corresponding observations o. V_t indicates the target node and let $V_s \in \mathrm{Sen}(V_t, O)$ be the node under study with states $\nu(V_s) = \{\nu_s^1, \dots, \nu_s^{m+1}\}$. Assume the probabilities of V_s are updated by normalization of the complements. Then, for any state ν_t of V_t, any configuration π' of $\mathrm{pa}(V_s)$ and any $\nu_s \in \nu(V_s)$, such that $P(\nu_s|\pi') \neq 1$, we have that

$$P(\nu_t|o) = \frac{ax + b}{cx + d}, \tag{2.4}$$

where $x = P(\nu_s|\pi')$, and a, b, c, d are constants related to ν_s and π'.

Proof

Without loss of generality, assume that $\nu_s = \nu_s^{m+1}$. By the fundamental rule we get

$$P(\nu_t|o) = \frac{P(\nu_t, o)}{P(o)}. \tag{2.5}$$

Both the numerator and denominator of (2.5) are investigated, which yields the numerator and denominator of (2.4), respectively.

For abbreviation, let $\Lambda = \{V_1, \dots, V_n\} \backslash (\{V_t\} \cup O)$ and $I = \{1, \dots, n\}$.

Consider the numerator of (2.5), $P(\nu_t, o)$. From the chain rule and marginalization, this joint probability can be written as a sum of products of the networks conditional probabilities,

$$P(\nu_t, o) = \sum_{\Lambda} \left(\prod_{i \in I} p(V_i | \,\mathrm{pa}(V_i)) \right) \Bigg|_{\substack{V_t = \nu_t \\ O = o}}$$

Then,

$$P(\nu_t, o) = \sum_{\Lambda} \left(P(V_s | \,\mathrm{pa}(V_s)) \prod_{i \in I \backslash \{s\}} P(V_i | \,\mathrm{pa}(V_i) \right) \Bigg|_{\substack{V_t = \nu_t \\ O = o}} \tag{2.6}$$

Let $g(x, V_s \cup pa(V_S)) = P(V_s | \operatorname{pa}(V_s))(x)$. The variable x of this theorem corresponds the variable x^* of Definition 2.2. Thus, we have for $s \in \Omega_{V_s}$ and $\pi \in \Omega_{\operatorname{pa}(V_s)}$

$$g(x, (s, \pi)) = \begin{cases} c(s, \pi) & \text{if } \pi \neq \pi' \\ \dfrac{c(s, \pi)(1 - x)}{1 - c(\nu_s^{m+1}, \pi)} & \text{if } \pi = \pi' \wedge s \neq \nu_s^{m+1} \\ x & \text{if } \pi = \pi' \wedge s = \nu_s^{m+1} \end{cases}$$

where $c(V_s \cup \operatorname{pa}(V_s))$ corresponds to the original $P(V_s | \operatorname{pa}(V_s))$ of the Bayesian network. In other words, $g(x, V_s \cup \operatorname{pa}(V_s)) = h_1(V_s \cup \operatorname{pa}(V_s)) + x \cdot h_2(V_s \cup \operatorname{pa}(V_s))$. Rewriting (2.6) gives

$$P(\nu_t, o)(x) = \sum_{\Lambda} g(x, V_s \cup \operatorname{pa}(V_s)) \prod_{i \in I \setminus \{s\}} g_i(V_i \cup \operatorname{pa}(V_i)) \Big|_{\substack{V_t = \nu_t \\ O = o}}$$

By Lemma 2.4 we get $P(\nu_t, o)(x) = ax + b$. Similar considerations yield that also $P(o)$ only depends linearly on x. This concludes the proof.

□

2.2.1 Topological Considerations

To prepare the ground for topological considerations in the case of influence diagrams, we investigate the flow of the probability potentials during absorptions in the junction tree.

Theorem 2.6

Let J be the junction tree of the investigated Bayesian network with all separators equal to one. Then, for any $C \in J$ and $s \in \Omega_S$

$$\phi_S(s) = \sum_{C \setminus S} \phi'_C = a_1(s)x + a_0(s) \tag{2.7}$$

where ϕ'_C denotes the probability potential of C after all its children in J have been absorbed into C as part of a collect towards the root. S is the separator between C and the parent of C in J. If C is the root then $S = \emptyset$.

Proof

Denote by $\Lambda_J(C)$ the subtree in J, which has C as root. We know $\phi'_C = P(C, o')$, where $O' = o'$ is the part of the evidence $O = o$ entered into cliques in $\Lambda(C)$. Hence,

$$\phi_S(s) = \sum_{C \setminus S} P(C, o') = \sum_{C \setminus S} \sum_{V(G) \setminus C} \prod_{V_i \in A(C)} P(V_i | \operatorname{pa}(V_i)) \Big|_{O' = o'},$$

where $A(C_i)$ denotes all the variables, V_i, which have $P(V_i|\,\mathrm{pa}(V_i))$ associated to a clique in $\Lambda_J(C)$. Similar to the proof of Theorem 2.5, Lemma 2.4 can be applied to get the desired result. We abstain from repeating the details.

□

We only consider the situation after a collect, since we will need this result in our work for influence diagrams. Note that the linearity of $\phi_S(s)$ after both a collect and a distribute also is of the form $a(s)x+b(s)$. In the distribute phase, $\phi_S(s) = P(S,o)$, which can be calculated by the chain rule as in the earlier proofs.

2.3 Sensitivity Analysis With Multiple Nodes Under Study

In this section we will give the proof of linearity in the case with more than one conditional probability under study. In this case we can have conflicts when two conditional probabilities under study require two different states of the set of parents for the same variable. For example, if $x_1 = P(V_{s_i} = \nu^1_{s_i}|\pi'_i)$ and $x_2 = P(V_{s_i} = \nu^1_{s_i}|\pi''_i)$ then the coefficient of x_1x_2 is zero. But, if $\pi'_i = \pi''_i$ the setup is inconsistent because $x_1 = x_2$ and then x_1 and x_2 cannot be ranged independently.

With this setup the target probability becomes a general polynomial where each variable only occurs linearly, which we will show in the following theorem. In the following $\mathcal{P}(S)$ denotes the power set of a set S.

Theorem 2.7

Let $B = (G,P)$ be a Bayesian network and $O \subseteq V(G)$ be the set of observed nodes with the corresponding observations o. Let V_t be the target node and $Sen(V_t,O)$ the sensitivity set for V_t given O. For any value ν_t of V_t and any set of nodes $\{V_{s_1},\ldots,V_{s_m}\} \subseteq Sen(V_t,O)$ under study, updated by normalization of the complements, we have that

$$P(\nu_t|o) = \frac{\displaystyle\sum_{M\in\mathcal{P}(\{1,\ldots,m\})}\left(a_M\prod_{i\in M}x_i\right)}{\displaystyle\sum_{M\in\mathcal{P}(\{1,\ldots,m\})}\left(b_M\prod_{i\in M}x_i\right)}, \tag{2.8}$$

where $x_i = P(\nu_{s_i}|\pi'_i) \neq 1$, $i \in \{1,\ldots,m\}$. Each a_M and b_M is a constant related to the values ν_{s_i} of V_{s_i} and π'_i of $\mathrm{pa}(V_{s_i})$.

Proof

By using the fundamental law the target probability $P(\nu_t|o)$ equals

$$P(\nu_t|o) = \frac{P(\nu_t,o)}{P(o)}.$$

Consider the nodes under study $V_{s_1}, \dots, V_{s_m} \in Sen(V_t, O)$. Without loss of generality, the sensitivity of the target probability will be investigated with regard to the conditional probabilities, $x_1 = P(\nu_{s_1}|\pi'_1), \dots, x_m = P(\nu_{s_m}|\pi'_m)$, where each π'_i is a specific configurations of the states of the nodes in $\mathrm{pa}(V_{s_i})$. Note, these configurations can yield to contradictions, as discussed in the beginning of this section.

As in the proof of Theorem 2.5 we only show our claim for $P(\nu_t, o)$, similar considerations can then be performed for $P(o)$. The proof now proceeds by induction in m.

Let $m = 1$. In this case the numerator of (2.8) reduces to $P(\nu_t, o)(x_1) = a_{\{1\}}x_1 + a_{\emptyset}$. Since we updated the probabilities according to normalization of complements, this statement is true due to Theorem 2.5.

Now, assume that the theorem is true for $m = k$, i.e.

$$P(\nu_t, o)(x_1, \dots, x_k) = \sum_{M \in \mathcal{P}(\{1,\dots,k\})} \left(a_M \prod_{i \in M} x_i \right). \tag{2.9}$$

We would like to show that it also holds for $m = k + 1$. By Theorem 2.5 we get

$$P(\nu_t, o)(x_1, \dots, x_{k+1}) = a(x_1, \dots, x_k)x_{k+1} + b(x_1, \dots, x_k). \tag{2.10}$$

Furthermore, we can use (2.9) to obtain

$$P(\nu_t, o)(x_1, \dots, x_{k+1}) = \sum_{M \in \mathcal{P}(\{1,\dots,k\})} \left(a_M(x_{k+1}) \prod_{i \in M} x_i \right) \tag{2.11}$$

where $a_M(x_{k+1})$ denotes that the coefficients are a function of x_{k+1}. Combining (2.10) with (2.11) and forming the partial derivative with respect to x_{k+1} yields

$$\frac{\partial P(\nu_t, o)(x_1, \dots, x_k)}{\partial x_{k+1}} = a(x_1, \dots, x_k) = \sum_{M \in \mathcal{P}(\{1,\dots,k\})} \left(\frac{\partial a_M(x_{k+1})}{\partial x_{k+1}} \prod_{i \in M} x_i \right).$$

Since $a(x_1, \dots, x_k)$ does not depend on x_{k+1} for all $x_1, \dots, x_k$, it must hold, that

$$\forall M \in \mathcal{P}(\{1, \dots, k\}) : \frac{\partial a_M(x_{k+1})}{\partial x_{k+1}} = c_M,$$

for some constant c_M. Thus, $\forall M \in \mathcal{P}(\{1, \dots, k\})$,

$$\int \frac{\partial a_M(x_{k+1})}{\partial x_{k+1}} \,\mathrm{d}x_{k+1} = \int c_M \;\mathrm{d}x_{k+1} = c_M x_{k+1} + d_M.$$

In other words, $a_M(x_{k+1}) = c_M x_{k+1} + d_M$, therefore, it is linear in x_{k+1}. Inserting this expression in (2.11) and multiplying gives

$$\begin{aligned} P(\nu_t, o)(x_1, \dots, x_{k+1}) &= \sum_{M \in \mathcal{P}(\{1,\dots,k\})} \left((c_M x_{k+1} + d_M) \prod_{i \in M} x_i \right) \\ &= \sum_{M \in \mathcal{P}(\{1,\dots,k+1\})} \left(a'_M \prod_{i \in M} x_i \right) \end{aligned}$$

This corresponds to the desired form of (2.8) and thus concludes our proof.

□

For abbreviation, we leave out the set notation for the indexes of the coefficients, in situations where this causes no ambiguities. For example, $a_{\{1,2\}}$ is written as a_{12} and $a_\emptyset$ as a_0.

2.4 Calculating The Coefficients

We now only need to show how to calculate the coefficients in the expression for $P(\nu_t|o)(\bar{x})$. Earlier we showed

$$P(\nu_t|o)(\bar{x}) = \frac{P(\nu_t, o)(\bar{x})}{P(\nu_t|o)(\bar{x})} = \frac{\sum_{M \in \mathcal{P}(\{1,\dots,m\})} \left(a_M \prod_{i \in M} x_i \right)}{\sum_{M \in \mathcal{P}(\{1,\dots,m\})} \left(b_M \prod_{i \in M} x_i \right)}. \tag{2.12}$$

Let us start by determining the coefficients of $P(\nu_t, o)(\bar{x})$. To determine these 2^m coefficients it is necessary to have 2^m values for $P(\nu_t, o)(\bar{x})$. The question is how to choose the 2^m values $\bar{v} = \bar{x}$ to insert, such that we obtain the wanted solution in an easy way. Once the values are chosen we insert each $\bar{v}_i$ into the network and propagate using e.g. Hugin. We can then read $y_i = P(\nu_t, o)(\bar{v}_i)$.

For $m = 2$ we have

$$\begin{aligned} a_{12}v_{11}v_{12} + a_1 v_{11} + a_2 v_{12} + a_0 &= y_1 \\ a_{12}v_{21}v_{22} + a_1 v_{21} + a_2 v_{22} + a_0 &= y_2 \\ a_{12}v_{31}v_{32} + a_1 v_{31} + a_2 v_{32} + a_0 &= y_3 \\ a_{12}v_{41}v_{42} + a_1 v_{41} + a_2 v_{42} + a_0 &= y_4 \end{aligned} \tag{2.13}$$

Choosing $v_{4*} = (0,0)$, $v_{3*} = (0,1)$, $v_{2*} = (1,0)$, and $v_{1*} = (1,1)$ the above reduces to the following linear equation system

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{12} \\ a_1 \\ a_2 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \quad \Leftrightarrow \quad Ca = y$$

which is solved easily using e.g. the Gauss-Jordan method, [Fraleigh and Beauregard, 1990, p.25]. Hence the equation system is consistent and has a unique solution, $a_0 = y_1$, $a_1 = y_2 - a_0$, $a_2 = y_3 - a_0$, and $a_{12} = y_4 - a_1 - a_2 - a_0$. Thus, letting $\bar{v}$ obtain all 2^m corner points of the m-cube $\partial[0,1]^m$ we obtained a simple linear equation system to solve.

The same method as shown above can be used for the general case with m conditional probabilities under study. In this case C is a $2^m \times 2^m$ matrix, a is a $2^m \times 1$ matrix, and y is a $2^m \times 1$ matrix. The values of $\bar{v}_i$ are again generated as the 2^m corner points of $\partial[0,1]^m$. More formally, these corner points are given by the following formula

$$V = \left\{ \bar{v} \in \mathbb{R}^m \;\middle|\; \exists M \in \mathcal{P}(\{1,\dots,m\}), l \in \{0,1\} : \bar{v} = \sum_{i \in M} l \cdot \bar{e}_i \right\},$$

where $\{\bar{e}_1, \dots, \bar{e}_m\}$ is the standard basis of $\mathbb{R}^m$.

The above method determines all the 2^m coefficients in $P(\nu_t, o)(\bar{x})$. Using a similar procedure, the coefficients in $P(\nu_t, o)(\bar{x})$ can be found. Thus, a total of $2 \cdot 2^m$ propagations are needed to determine all the coefficients in $P(\nu_t|o)(\bar{x})$

A small optimization can be achieved if there exists a conflict between the parent configurations. Let $W = \mathrm{pa}(V_{s_i}) \cap \mathrm{pa}(V_{s_j}) \neq \emptyset$ and denote by $W(\pi')$ the configurations in π' regarding W. Thus, if $W(\pi'_i)$ is in conflict with $W(\pi'_j)$, we know that they cannot coexist and therefore are the a_M coefficient of all terms containing $x_i x_j$ equal to zero. Hence, $2 \cdot 2^{m-2}$ coefficients need not to be determined, which saves equally many propagations. Also if $(V_{s_j} \cup \mathrm{de}(V_{s_j})) \cap O = \emptyset$, we know that all the b_M coefficients containing j are zero.

The above method of finding the coefficients is in our opinion more applicable than the one mentioned in [Coupé and van der Gaag, 1998b] for $m = 1$. Here they try to find the coefficients of $P(\nu_t|o)(x)$ by reading $y_i = (\nu_t|o)(v)$ and thus solving for the a and b coefficients simultaneously. To avoid homogeneous linear equation systems with only trivial solutions they have to divide the expression $\frac{ax+b}{cx+d}$ by c such that there are only three parameters to determine. This is complicated to apply since it is necessary to ensure $c \neq 0$ and the probabilities in the network could always make c zero.

2.5 Summary and Future Research

The results from [Coupé and van der Gaag, 1998b] were extended to both non-binary variables and several conditional probabilities under study. Thus, we are now able to find an analytical expression for $P(v_t|o)(\bar{x})$ for any Bayesian network. Unfortunately, the number of propagations needed to determine the coefficients of the function were exponential in the number of parameters under study.

One should look into whether exploiting the structure of the junction tree can reduce the needed number of propagations. For example, in the case of one probability under study, an alternative way could be to insert $V_t = \nu_t$ as evidence and collect towards the clique C_{V_s} containing $P(V_s|\,\mathrm{pa}(V_s))$, without multiplying $P(V_s|\,\mathrm{pa}(V_s))$. This propagation results in a table $T_{C_{V_s}}(c)$ at C_{V_s} which we store. A similar calculation is performed where $V_t = \nu_t$ is not entered as evidence, this results in the table $T'_{C_{V_s}}(c)$ which we also store. We can now calculate $P(v_t|o)(v)$ by

$$P(v_t|o)(v) = \frac{P(v_t, o)(v)}{P(o)(v)} = \frac{P(V_s|\,\mathrm{pa}(V_s))_{x=v} * T_{C_{V_s}}(c)}{P(V_s|\,\mathrm{pa}(V_s))_{x=v} * T'_{C_{V_s}}(c)}$$

Notice how all the multiplications happen locally. All together, we only needed two global propagations and eight local multiplications to perform the insertion scheme described in Section 2.4. In large networks this is definitely an optimization. When the work on this project is continued, one should investigate how this structure can be generalized to multiple probabilities under study.

Chapter 3

Sensitivity Analysis in Influence Diagrams

This chapter describes how to perform sensitivity analysis in influence diagrams when the unreliable parameters are found in the conditional probabilities of the chance nodes. We are interested in analyzing how the next decision to be taken depends upon these parameters.

3.1 Motivation

Consider the following scenario. A decision-maker has to decide whether to dismiss or keep an employee based on the result of a drug test. Evaluating the influence diagrams with the current values of the parameters might suggest a dismissal, but sensitivity analysis tells that within the bounds of imprecision the alternating decision could have been made as well. Under these circumstances the decision-maker might prefer not to dismiss the employee. Thus, sensitivity analysis reveals, whether a decision is affected by the imprecisions and if yes it enables the decision-maker to let an ethical preference — not reflected in the utility values — control the decision.

3.2 A Coarse Concept of Sensitivity Analysis in Influence Diagrams

In the description of influence diagrams we use the notation found in [Jensen *et al.*, 1994]. Thus, influence diagrams with $U = U_D \cup U_R$ where $|U_D| = n$ and $|U_R| = m$ are considered. Our attention now concerns the next decision, D_η, to be taken in the diagram. It is given by the outermost decision, which has all its information variables instantiated.

Suppose that the k parameters under study are located in the nodes $V_s = \{V_{s_1}, ..., V_{s_k}\} \subseteq U_R$. Since the parameters are conditional probabilities they are consequently denoted $x_i = P(\nu_{s_i}|\pi'_i)$, where $i = 1, \ldots, k$, $\nu_{s_i} \in \nu(V_{s_i})$ and $\mathrm{pa}(V_{s_i}) = \pi'_i$ is a specific configuration of parents. Note, if $V_{s_j} = V_{s_k}$ and $\nu_{s_i} = \nu_{s_j}$ then $\pi'_j \neq \pi'_k$. Let $\bar{x} = (x_1, \ldots, x_k)$.

The uncertainty on the parameters is modeled by assuming, that the exact value of $\bar{x}$ is given by a multivariate normal distribution with mean value $\bar{\mu}$ and covariance matrix Σ. Since each x_i denotes a probability, we have $x \in [0, 1]^k$. Thus, $\bar{g} = \bar{\mu}$ is entered into the network as the best guess on the values in $\bar{x}$. The aim of sensitivity analysis can now be described as follows. Select a certainty c and determine a confidence interval X, such that $P(\bar{x} \in X) = c$. Since $\bar{x}$ follows a multivariate Gaussian distribution, the shape of X will be a k dimensional ellipsoid intersected by $[0,1]^k$. Let d_g be the decision taken in $\bar{g}$.

$$d_g = \arg\max_{d \in \nu(D_\eta)} \left(\underset{D_\eta = d}{\mathrm{EU}}(\bar{g}) \right)$$

Would we have taken another decision if $\bar{x}$ had been varied within the bounds of the confidence interval? That is

$$\exists d' \in \nu(D_\eta) \backslash \{d_g\} : \exists \bar{x} \in X : \arg\max_{d \in \nu(D_\eta)} \left(\underset{D_\eta = d}{\mathrm{EU}}(\bar{x}) \right) = d'$$

At this point we perform some rather coarse restrictions. In order to simplify the general situation we expect the uncertainties of the parameters to be uncorrelated. This assumption is acceptable, but rarely the case. Furthermore, we assert that all the uncertainties have the same variance r_{V_s}. This situation is not very likely to occur in reality, but has to do for now. Thus, the confidence interval $X = B^k(\bar{g}, r_B) \cap [0,1]^k$, where $B^k(\bar{g}, r_{V_s})$ is a k-ball centered at $\bar{g}$ with radius $r_B = q \cdot r_{V_s}$, where q is a constant determined by c. As a tool in our sensitivity analysis we define the following concept.

Definition 3.1 (Radius of change)
The minimum Euclidean distance, denoted r_c, from $\bar{g} \in [0,1]^k$ to $\bar{p} \in [0,1]^k$, where a different decision than d_g is selected, is defined as

$$\begin{aligned} r_c \;=\; & \min(\{r \mid \exists \bar{p} \in (\partial B^k(\bar{g}, r) \cap [0,1]^k) \;\Rightarrow \\ & \qquad \exists d'_g \in \nu(D_\eta) \backslash \{d_g\} : \underset{D_\eta = d'_g}{\mathrm{EU}}(\bar{p}) = \underset{D_\eta = d_g}{\mathrm{EU}}(\bar{g})\}), \end{aligned} \tag{3.1}$$

where ∂B^k denotes the border of B^k. If no r fulfills (3.1) we define $r_c = \infty$.

Since the expected utility, as a function of $\bar{x}$, is continuous, it is sufficient in the above definition to require that $\mathrm{EU}_{D_\eta = d'_g}(\bar{p}) = \mathrm{EU}_{D_\eta = d_g}(\bar{g})$ instead of $\geq$.

If $r_B < r_c$ the decision is unaffected by the modeled uncertainties in the parameters, see Figure 3.1. A more practical application of r_c occurs in situations where the decision-maker does not know r_B. By simply considering r_c an understanding of the sensitivity of the decision with respect to the insecure parameters can be obtained. The greater the value of r_c the less sensitive is the decision to variations in the parameters. Should the decision-maker become aware of the true elliptic shape of the confidence interval X, then if $X \backslash B^k(\bar{g}, r_c) = \emptyset$ the decision is insensitive to the uncertainties in the parameters.

In the following sections we will only consider binary influence diagrams, i.e. all random variables have two states and all decision variables have two decision alternatives. To calculate

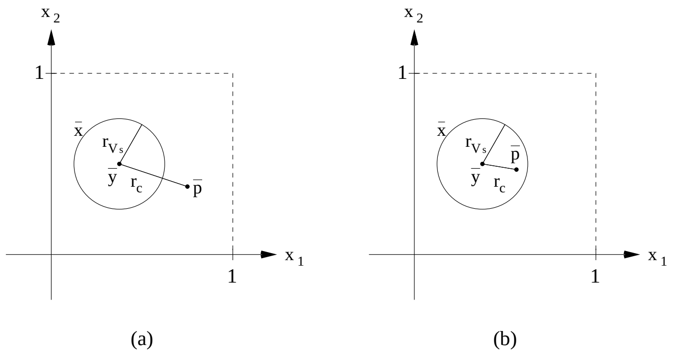


Figure 3.1: Illustration of when the decision is affected by the modeled uncertainties in the parameters. (a) does not yield a problem, because $r_c > r_B$, and is therefore unaffected. But, (b) is problematic, because $r_c < r_B$.

r_c in binary influence diagrams, it is sufficient to consider the set of intersection points, I_p, between $\mathrm{EU}_{D_1=d_1}(\bar{x})$ and $\mathrm{EU}_{D_1=\neg d_1}(\bar{x})$. That is

$$I_p = \{\bar{x} | \underset{D_1=d_1}{\mathrm{EU}}(\bar{x}) - \underset{D_1=\neg d_1}{\mathrm{EU}}(\bar{x}) = 0\}. \tag{3.2}$$

In this case r_c equals the minimum distance from $\bar{g}$ to a point in $I_p \cap [0,1]^k$. Again, if $I_p \cap [0,1]^k = \emptyset$, $r_c = \infty$.

Now, assume that the target decision variable is non-binary, i.e. it has more than two states. Let D_i have the decision alternatives $\nu(D_i) = \{d_i^1, \ldots, d_i^k\}$, and let d_i^g be the decision alternative taken with $\bar{x} = \bar{g}$. Now, we compare the expected utility of $D_i = d_i^g$ with the expected utilities of all the other decision alternatives, $d_i^1, \ldots, d_i^{g-1}, d_i^{g+1}, \ldots, d_i^k$. For all of these $k-1$ comparisons of two decision alternatives we find the radius of change, then, the smallest of these radii of changes is the radius of change for D_i. Thus, once a method has been developed for comparing two decision alternatives, the non-binary case can also be solved.

3.3 Radius of Change in Influence Diagrams with One Decision

In the following we only consider influence diagrams with one decision, thus, $n = 1$. Therefore, $D_\eta = D_1$, which requires I_0 to be instantiated. Let i_o be the observed values of I_0 and assume $V_s \subseteq I_1$. Thus, to find the maximum expected utility in D_1, we must look at ρ_1, which yields

$$\underset{D_1}{\mathrm{MEU}} \;=\; \rho_1 \;=\; \max_{D_1} \sum_{I_1} P(I_1 | I_0, D_1) * \psi.$$

Hence, the expected utility for $d \in \nu(D_1)$ is given by

$$\begin{aligned}
\underset{D_1=d}{\mathrm{EU}}(\bar{x}) &= \sum_{I_1} P(I_1|I_0, D_1) * \psi \Big|_{\substack{I_0=i_0\\D_1=d}} \\
&= \frac{1}{P(I_0)} \sum_{I_1} P(I_0, I_1|D_1) * \psi \Big|_{\substack{I_0=i_0\\D_1=d}} \\
&= \frac{1}{P(I_0)} \left(\sum_{I_1} \prod_{i=1,\dots,m} P(V_i|\,\mathrm{pa}(V_i)) * \psi \right) \Big|_{\substack{I_0=i_0\\D_1=d}}
\end{aligned}$$

The utility function ψ does not depend on x_i, thus, multiplying it will not change the result of Theorem 2.7. Hence,

$$\underset{D_1=d}{\mathrm{EU}}(\bar{x}) = \sum_{M \in \mathcal{P}(\{1,\dots,k\})} \left(a_M(d) \prod_{i \in M} x_i \right), \tag{3.3}$$

where $a_M(d)$ now also depends on the values of ψ and the configuration i_0. Inserting this expression into (3.2) yields $I_p = \{\bar{x}|F(\bar{x}) = 0\}$, where

$$F(\bar{x}) = \sum_{M \in \mathcal{P}(\{1,\dots,k\})} \left((a_M(d) - a_M(\neg d)) \prod_{i \in M} x_i \right) \tag{3.4}$$

One evaluation of an influence diagram yields both $\mathrm{EU}_{D_1=d_1}(\bar{x})$ and $\mathrm{EU}_{D_1=\neg d_1}(\bar{x})$, therefore the $2 \cdot 2^k$ constants in (3.4) can be found by 2^k evaluations.

The square of the distance from a random point $\bar{x} \in \mathbb{R}^k$ to $\bar{g}$ is defined as

$$\mathrm{dist}(\bar{g}, \bar{x}) = (\bar{g} - \bar{x}) \cdot (\bar{g} - \bar{x}). \tag{3.5}$$

Now, picture a situation like the one sketched in Figure 3.2(a) simplified to two dimensions, which corresponds to two parameters under study. Our goal is to find the minimum distance from $\bar{g}$ to an $\bar{x}$ in the part of I_p which is located inside the universe $[0,1]^k$, i.e. $r_c^2 = \min \mathrm{dist}(\bar{g}, I_p \cap [0,1]^k)$. Using a divide and conquer strategy, this problem is split into two subproblems. First, we solve the problem of finding all extrema of the function $\mathrm{dist}(\bar{g}, I_p)$. This gives us a set of candidate points on F. Since we are only interested in the extrema located in the universe $[0,1]^k$, we would like to discard those laying outside of $[0,1]^k$. But, we need to do this with care, because the minimum distance could have been found at the closest intersection point of $F(\bar{x})$ with $\partial[0,1]^k$, see Figure 3.2(b). By finding $I_p \cap \partial[0,1]^k$ and adding it to the set of candidates we are certain to cover the above situation.

In order to solve the extrema problem we note that both $F(\bar{x})$ and $\mathrm{dist}(\bar{g}, \bar{x})$ have continous

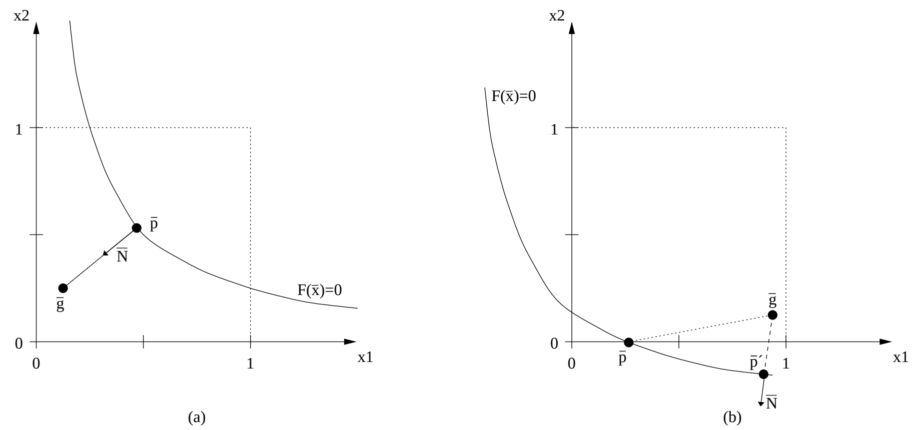


Figure 3.2: Finding $\mathrm{dist}(\bar{g}, I_p \cap [0,1]^k)$ in the two-dimensinal case. In (a) the point with minimum distance is an extrema of $\mathrm{dist}(\bar{k}, I_p \cap [0,1]^k)$, whereas in (b) the minimum is located on $\partial[0,1]^2$.

first-order partial derivatives,

$$\begin{aligned} \frac{\partial F(\bar{x})}{\partial x_j} &= \sum_{M \in \mathcal{P}(\{1,\ldots,k\})} \left(a_M \cdot I \prod_{i \in M \setminus \{j\}} x_i \right) \\ \frac{\partial\, \mathrm{dist}(\bar{g}, \bar{x})}{\partial x_j} &= -2(g_j - x_j) \end{aligned}$$

where $I = 1$ if $j \in M$ and $I = 0$ otherwise. These derivatives ensure that finding $\min \mathrm{dist}(\bar{g}, I_p)$ can be done by using the method of Lagrange multipliers [Edwards Jr. and Penny, 1994, p.807ff]. Define

$$\nabla_0 = \{\bar{p} \mid \nabla F(\bar{p}) = \bar{0} \wedge F(\bar{p}) = \bar{0}\} \tag{3.6}$$

$$E = \{\bar{p} \mid \nabla F(\bar{p}) \neq \bar{0} \wedge F(\bar{p}) = \bar{0} \wedge \exists \lambda \in \mathbb{R} : \nabla\, \mathrm{dist}(\bar{g}, \bar{p}) = \lambda \nabla F(\bar{p})\} \tag{3.7}$$

The method of Lagrange multipliers states that the minimum value of dist, subject to the condition $F(\bar{x}) = 0$, occurs among $\nabla_0 \cup E$. Determining E corresponds to defining

$$L(\bar{x}, \lambda) = \mathrm{dist}(\bar{g}, \bar{x}) - \lambda F(\bar{x})$$

and then solving the $k+1$ equations $\frac{\partial L(\bar{x},\lambda)}{\partial \lambda} = 0$, $\frac{\partial L(\bar{x},\lambda)}{\partial x_1} = 0$, $\ldots$, $\frac{\partial L(\bar{x},\lambda)}{\partial x_k} = 0$, see [Fraleigh, 1985, p.745ff]. Solving such a system of equations can be done efficiently with numeric methods or be done by symbolic computation tools, such as Maple.

The second task corresponds to finding

$$B = \{\bar{x} | \bar{x} \in \partial[0,1]^k \wedge F(\bar{x}) = 0\} \tag{3.8}$$

Consequently, if $\min(\emptyset) \triangleq \infty$, then $r_c^2 = \min(\{\mathrm{dist}(\bar{g}, \bar{p}) | \bar{p} \in (((\nabla_0 \cup E) \cap [0,1]^k) \cup B)\})$.

In other words, a set of candidates which is considerably smaller than $I_p \cap [0,1]^k$ is generated. This method does not deliver any solutions, but is applicable for an algorithmic implementation, e.g., in Maple. For $k = 1$ and $k = 2$ the above sets can be evaluated, which is done in Sections 3.3.1 and 3.3.2. But, already with $k = 2$ an explicit expression for r_c becomes quite cumbersome to formulate.

3.3.1 Radius of Change with One Parameter Under Study

With this setup $k = 1$. Thus,

$$\begin{aligned}
F(x_1) &= a_1 x_1 + a_0, \\
\mathrm{dist}(g_1, x_1) &= (g_1 - x_1)^2, \\
\frac{\partial L(x_1,\lambda)}{\partial x_1} &= -2(g_1 - x_1) - \lambda a_1, \\
\frac{\partial L(x_1,\lambda)}{\partial \lambda} &= -(a_1 x_1 + a_0).
\end{aligned}$$

To determine a_1 and a_0 we evaluate the influence diagram with

$$\begin{aligned}
x = 0 \;\Rightarrow\; \underset{D_1=d_1}{\mathrm{EU}}(x_1) - \underset{D_1=\neg d_1}{\mathrm{EU}}(x_1) &= a_0 \\
x = 1 \;\Rightarrow\; \underset{D_1=d_1}{\mathrm{EU}}(x_1) - \underset{D_1=\neg d_1}{\mathrm{EU}}(x_1) &= a_1 + a_0
\end{aligned}$$

An analytical expression for r_c can be determined in the following way. Since $\forall x_1 : \exists \lambda : -2(g_1 - x_1) = \lambda a_1$ does not give further restrictions on x_1, only on λ given x_1 which we are not interested in, therefore, the λ-term in E is left out. Hence, the sets ∇_0, E and B are

$$\begin{aligned}
\nabla_0 &= \{x_1 | a_1 = 0 \land a_0 = 0\} \\
E &= \{x_1 | a_1 \neq 0 \land a_1 x_1 + a_0 = 0\} \\
B &= \{x_1 | x_1 \in [0,1] \land a_1 x_1 + a_0 = 0\}
\end{aligned}$$

Now,

$$\nabla_0 = \begin{cases} \mathbb{R} & \text{if } a_1 = 0 \land a_0 = 0 \\ \emptyset & \text{otherwise} \end{cases}$$

$$E = \begin{cases} \{-\frac{a_0}{a_1}\} & \text{if } a_1 \neq 0 \\ \emptyset & \text{otherwise} \end{cases}$$

$$B = \begin{cases} \{-\frac{a_0}{a_1}\} & \text{if } a_1 \neq 0 \\ \{x_1 \mid 0 \leq x_1 \leq 1\} & \text{if } a_0 = 0 \land a_1 = 0 \\ \emptyset & \text{otherwise} \end{cases}$$

Note that $\min \mathrm{dist}(g_1, \mathbb{R} \cap [0,1]) = 0$. By combining all this information an analytical expression for r_c is achieved.

$$r_c = \begin{cases} |g - \frac{a_0}{a_1}| & \text{if } a_1 \neq 0 \wedge -\frac{a_0}{a_1} \in [0,1] \\ 0 & \text{if } a_1 = 0 \wedge a_0 = 0 \\ \infty & \text{otherwise} \end{cases}$$

3.3.2 Radius of Change with Two Parameters Under Study

With this setup $k = 2$, and

$$\begin{aligned} \mathrm{dist}(\bar{g}, \bar{x}) &= (g_1 - x_1)^2 + (g_2 - x_2)^2 \\ F(x_1, x_2) &= a_{12}x_1x_2 + a_1x_1 + a_2x_2 + a_0 \end{aligned}$$

Thus, $F(x_1, x_2) = 0$ corresponds to a general second degree equation, i.e. a cone intersection. Since it contains no quadratic terms, $F(\bar{x}) = 0$ will form a (possibly degenerated) hyperbola. The transverse axis of the hyperbola will be rotated 45 degrees as in Figure 3.3, see [Edwards Jr. and Penny, 1994, p.562ff, p.567ff].

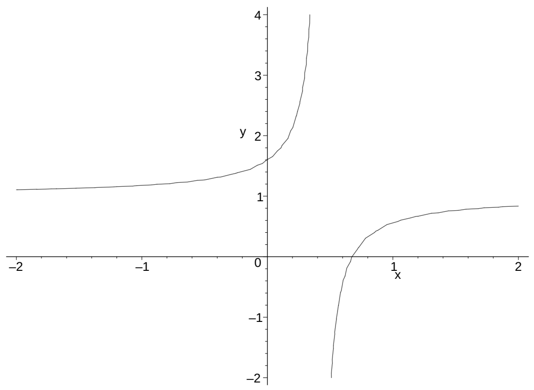


Figure 3.3: An example of a hyperbola, for which the transverse axis is rotated 45 degrees.

To determine the constants we evaluate the influence diagram with the following four setups.

$$\begin{aligned} \{x_1 = 0, x_2 = 0\} &\Rightarrow a_0, \\ \{x_1 = 1, x_2 = 0\} &\Rightarrow a_0 + a_1, \\ \{x_1 = 0, x_2 = 1\} &\Rightarrow a_0 + a_2, \\ \{x_1 = 1, x_2 = 1\} &\Rightarrow a_{12} + a_2 + a_1 + a_0. \end{aligned}$$

It is now necessary to determine the sets ∇_0, E and B.

$$\begin{aligned}\nabla_0 &= \{(x_1,x_2)|a_{12}x_2+a_1=0\wedge a_{12}x_1+a_2=0\wedge F(x_1,x_2)=0\}\\ &= \begin{cases}\mathbb{R}^2 & \text{if } (a_{12}\neq 0\wedge -\frac{a_1a_2}{a_{12}}+a_0=0)\quad\vee\\ & \qquad (a_1=0\wedge a_2=0\wedge a_0=0\wedge a_{12}=0)\\ \emptyset & \text{otherwise}\end{cases}\end{aligned}$$

In $\mathbb{R}^2$, $\nabla\operatorname{dist}(\bar{g},\bar{x})=\lambda\nabla F(\bar{x})$ is the same as solving $\det(\nabla F(\bar{x}),\nabla\operatorname{dist}(\bar{g},\bar{x}))=0$, [Fraleigh and Beauregard, 1990, p.197ff]. Hence,

$$E = \left\{(x_1,x_2)|\begin{vmatrix}a_{12}x_2+a_1 & -2(g_1-x_1)\\ a_{12}x_1+a_2 & -2(g_2-x_2)\end{vmatrix}=0\wedge F(x_1,x_2)=0\wedge\nabla F(x_1,x_2)\neq 0\right\}$$

Solving the above equation system yields two 4th degree polynomials, A and B. The x_1 solutions are given as the roots of A whereas the x_2 solutions are given by the roots of $\frac{A}{B}$. The non complex roots correspond to the desired extrema points. We will abstain from showing the exact expression of the polynomials since they are rather extensive. We just note that once the values are entered, Maple will be able to find the roots.

Finally we consider the intersection of F with the border,

$$\begin{aligned}B &= \{(x_1,x_2)|(x_1=0\vee x_1=1\vee x_2=0\vee x_2=1)\wedge F(x_1,x_2)=0\}\cap[0,1]^2\\ &= \{(x_1,x_2)|(x_1=0\wedge a_2x_2+a_0=0)\vee(x_1=1\wedge a_{12}x_2+a_1+a_2x_2+a_0=0)\vee\\ &\qquad (x_2=0\wedge a_1x_1+a_0=0)\vee(x_2=1\wedge a_{12}x_1+a_1x_1+a_2+a_0=0)\}\cap[0,1]^2\end{aligned}$$

Depending on the coefficients, the solution to these four equations (in the above disjunction in B) yield a number of points or lines of type $x_i=B_i$, where $B_i\in\{0,1\}$. Since the transverse axis of the hyperbola is rotated 45 degrees, we know there will be only one intersection point on each edge of the square $[0,1]^2$, unless the hyperbola consists of the line $x_1=B_1$ and/or $x_2=B_2$. Finding the distance from such a line to $\bar{g}$ reduces to $\operatorname{dist}(\bar{g},x_i=B_i)=(g_{3-i}-B_i)^2$. Again, even though B can contain infinitely many points due to the lines, it is easy to find the minimum distance to the line.

The structure of ∇_0, E, B has now been shown, unfortunately the many cases spawned by various combinations of a_M equal to zero make it cumbersome to give an explicit expression for r_c. Instead the many cases motivate an implementation of the general algorithm, where $a_{12},\ldots,a_0$ are entered after which the relevant solutions are computed.

3.4 Cooper's Transformation and Utilities Under Study

In this section we present a method on how to transform sensitivity analysis in influence diagrams into sensitivity analysis in Bayesian networks. It serves as perspectivation of the theory developed in this chapter using the theory in Chapter 2. The method uses Cooper's Transformation to transform an influence diagram into a Bayesian network. Furthermore, this transformation makes it possible to have utilities under study.

3.4.1 Cooper's Transformation: A Calculation Trick to Convert Influence Diagrams into Bayesian Networks

The evaluation of influence diagrams is more complicated than the evaluation of Bayesian networks, therefore, an algorithm for converting a influence diagram into a Bayesian network is preferred, because this enables one to always use the newest propagation algorithm. Optimization of Bayesian network propagation algorithms is an active area of research, [Zhang, 1998].

Cooper initiated research in this direction in 1988, and therefore this calculation trick is called *Cooper's transformation*. Several improvements have been proposed by Shachter and Peot in 1992, [Zhang, 1998].

Definition 3.2 (Cooper's transformation)
Let I be an influence diagram with only one utility node, ψ, and decision variables, $D_1, \ldots, D_m$, where D_1 is the first decision and D_m is the last. Without loss of generality, assume that the utility function f_ψ is positive, if not a constant can be added such that it is. Now, ψ can be converted into a binary random variable with the following conditional probabilities

$$\begin{aligned} P(\psi = 1 |\, \mathrm{pa}(\psi)) &= \frac{f_\psi(\mathrm{pa}(\psi))}{M_\psi} \\ P(\psi = 0 |\, \mathrm{pa}(\psi)) &= 1 - P(\psi = 1 |\, \mathrm{pa}(\psi)), \end{aligned} \tag{3.9}$$

where $M_\psi = \max_{\mathrm{pa}(\psi)} \psi(\mathrm{pa}(\psi))$.

Next, each decision variable D_i is also converted into a random variable having the following conditional probability

$$P(D_i = \alpha_i |\, \mathrm{pa}(D_i)) = \frac{1}{|\nu(D_i)|}, \tag{3.10}$$

for each possible value α_i of D_i.

After this transformation, I is a Bayesian network denoted by I'.

The optimal decision rule δ_i^* for D_m can now be obtained by

$$\delta_i^*(\mathrm{pa}(D_m)) = \arg\max_{D_m} P_{I'}(D_m, \mathrm{pa}(D_m) | \psi = 1),$$

which can be evaluated using a inference algorithm for Bayesian networks, and the maximum expected utility for the optimal decision rule can be obtained by

$$\mathrm{MEU}(D_m) = \max_{D_m} P_{I'}(D_m, \mathrm{pa}(D_m) | \psi = 1) M_\psi .$$

Now, the conditional probability of D_m is changed to $P_{\delta_m^*}(D_k |\, \mathrm{pa}(D_m)$. Next, an optimal decision rule for D_{m-1} is computed in the same way. Thus, optimal decision rules for $D_m, \ldots, D_1$ are computed recursively in this manner.

Shacter and Peot's method reduces the evaluation of I into the following Bayesian network inference problems [Zhang, 1998]

$$P_{I'}(D_m, \mathrm{pa}(D_m)|\psi = 1), P_{I'}(D_{m-1}, \mathrm{pa}(D_{m-1})|\psi = 1), \ldots, P_{I'}(D_1, \mathrm{pa}(D_1)|\psi = 1).$$

Furthermore, [Zhang, 1998] shows that this evaluation can be performed faster than propagation in strong junction trees despite of the transformation overhead.

3.4.2 Cooper's Transformation and Sensitivity Analysis

We will know show that when applying Cooper's transformation on an influence diagram, I, then, performing a sensitivity analysis on I' is similar to performing sensitivity analysis on I, directly.

Assume that I only contains one decision, named D_1, and only one conditional probability under study, called x. Let V_s be the variable containing x. Thus, the expected utility in I' is given by

$$\mathop{\mathrm{EU}}_{d \in D_1}(x) = P_{I'}(D_1 = d, \mathrm{pa}(D_1)|\psi = 1) \cdot M_\psi.$$

Like in Section 3.3 we want to get rid of $\mathrm{pa}(D_1)$ in the above conditional probability. This can be done by dividing it by $T = P_{I'}(\mathrm{pa}(D_1)|\psi = 1)$, thus,

$$\begin{aligned}\mathop{\mathrm{EU}}_{d \in D_1}(x) &= P_{I'}(\mathrm{pa}(D_1)|\psi = 1)\frac{P_{I'}(D_1 = d, \mathrm{pa}(D_1)|\psi = 1)}{P_{I'}(\mathrm{pa}(D_1)|\psi = 1)} \cdot M_\psi \\ &= T \cdot P_{I'}(D_1 = d|\psi = 1) \cdot M_\psi,\end{aligned}$$

where T and M_ψ are constants with respect to x, assumed that $V_s \notin \mathrm{pa}(D_1)$, in which case $V_s \notin \mathrm{Sen}(D_1, \psi = 1)$.

Now, sensitivity analysis can be viewed as if it was for Bayesian networks. Since D_1 is the target node and the only observed nodes are $\mathrm{pa}(D_1)$, then V_s can be either an ancestor or a descendant of D_1.

First, assume that $V_s \in \mathrm{de}(D_1)$. Thus, $V_s \notin \mathrm{an}(\mathrm{pa}(D_1))$, and therefore, none of the descendants of V_s will be observed, see Figure 3.4(a). Hereby, [Coupé and van der Gaag, 1998b, Propostition 4.3] applies to the situation, thus,

$$P(D_1 = d|\psi = 1) = a(d)\,x + b(d),$$

and hence,

$$\mathop{\mathrm{EU}}_{d \in D_1}(x) = a'(d)\,x + b'(d),$$

where $a'(d)$ and $b'(d)$ also includes the constants T and M_ψ.

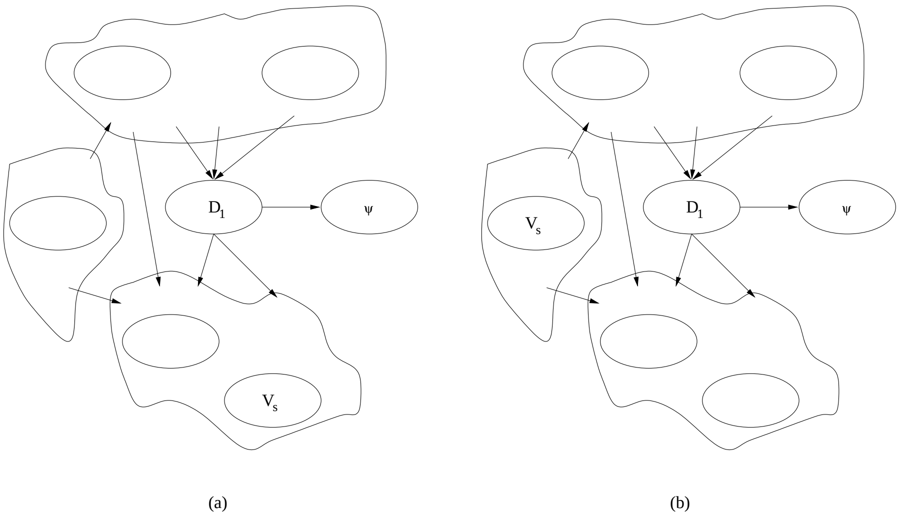


Figure 3.4: (a) The situation where $V_s \in \mathrm{de}(D_1)$. (b) The situation where $V_s \in \mathrm{an}(D_1)$.

Secondly, assume that $V_s \in \mathrm{an}(D_1)$. Then, $V_s \notin \mathrm{Sen}(D_1, \psi = 1)$, therefore, the expected utility is constant with respect to d, $\mathrm{EU}_{d \in D_1}(x) = c(d)$, see Figure 3.4(b). These results correspond to the results of Section 3.3.

Unfortunately, the use of Cooper's transformation when having more than one decision involves a lot of changes of tables when decisions change because of the ranging of the unknown parameters; therefore, the transformation approach is not suitable in the sense of sensitivity analysis.

3.4.3 Utilities Under Study Using Cooper's Transformation

We now shortly discuss how to implement sensitivity analysis when the node under study is a utility function. Here, we approach the situation by using the Cooper's transformation to deal with it in Bayesian networks.

In Bayesian networks it is not difficult to have utilities under study, because the utility node is converted into a random node, in which the sensitivity analysis can be performed using the methods described in Chapter 2. If the radius of change is desired, Cooper's transformation, described in Section 3.4.1, can be used using the method in Section 3.3.

As in the previous case, we have to keep in mind how the utility under study should be ranged. In this situation it is only ranged between the minimum and the maximum utility values in ψ. Generally, it may be desirable to range the utility beyond these bounds.

3.5 Summary

In this chapter we have looked into performing sensitivity analysis on influence diagrams with only one decision, the target decision. During this we made some rather coarse assumptions of the understanding of the confidence interval and we are primarily looking at uncertain parameters in the conditional probability tables. Although we can use Cooper's transformation to implement uncertain parameters in the utility nodes, too. Unfortunately, Cooper's transformation is not suitable when ranging one or more parameters because the decisions change during this procedure. Each time a decision changes tables need to be replaced, which is intractable.

We managed to find an explicit expression for the radius of change for sensitivity analysis with one conditional probability under study. But, already when having two under study, the expression was to cumbersome to express; therefore, a computer tool is needed, see Chapter 5. Before this theory is of any use, we need to be able to analyze diagrams with more than one decision.

The next subject to look at in this area of sensitivity analysis, is to look more deeply into the utilities under study, e.g. where the sensitivity analysis is implemented directly on influence diagrams.

Chapter 4

Influence Diagrams with Two Decisions

In this chapter we consider influence diagrams with two decisions. According to [Jensen *et al.*, 1994] the framework is thus $I_0 \prec D_1 \prec I_1 \prec D_2 \prec I_2$, where D_1 is the target decision. For the sake of simplicity, we assume $I_0 = \emptyset$, but our results are just as applicable if this is not the case.

In order to exploit the methods developed in the previous chapter of finding the radius of change between two decision alternatives using ∇_0, E and B, we need to find an expression for

$$F(\bar{x}) \;=\; 0, \quad \text{where } F(\bar{x}) \;=\; \underset{D_1=d_1}{\mathrm{EU}}(\bar{x}) - \underset{D_1=\neg d_1}{\mathrm{EU}}(\bar{x}).$$

It is therefore necessary to perform an analysis of $\mathrm{EU}_{D_1}(\bar{x})$ within the given setup. In the previous section this analysis was performed by looking at ρ_1, but in this chapter we connect the investigation to the structure of the strong junction tree instead. First we perform an investigation of the structure obtained with one conditional probability under study — this enables us to calculate r_c. Subsequently, the method to calculate r_c in the case of two probabilities under study is sketched.

4.1 Topological Considerations for One Probability Under Study

Let J be the strong junction tree of the influence diagram under investigation. In J we identify the following cliques.

- C_{D_1} is the clique containing D_1, closest to the root. Since $I_0 = \emptyset$, C_{D_1} will be the root, R, of J.
- C_{D_2} is the clique, closest to the root, containing D_2
- C_x is the clique to which $P(V_s|\,\mathrm{pa}(V_s))$ is assigned.

Without loss of generality, we assume $C_{D_1} \neq C_{D_2} \neq C_x$. Furthermore, $W_{(C_x,R)}$ denotes all the cliques in the path from C_x to R.

Now, let C be a clique in J with separator S between C and the parent of C, C_p, in J. If C is the root then $S = \emptyset$. By $\Lambda_J(C)$ we express the subtree in J that has C as root. Then the set

$$E(C) = \{V_i | \exists C_i \in (W_{(C_x,R)} \cap \Lambda_J(C)) : V_i \in C_i\} \backslash S$$

denotes all the variables from $W_{(C_x,R)}$ that have to be eliminated, before C can be absorbed into C_p, see Figure 4.1.

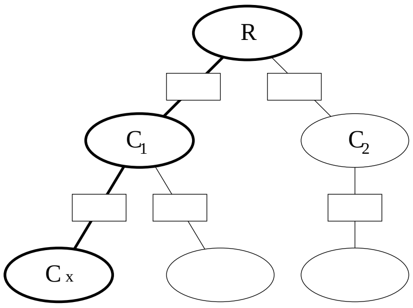


Figure 4.1: Illustration of the set E, here $W_{(C_x,R)} = \{C_x, C_1, R\}$. Since $\Lambda_J(C_2) \cap W_{(C_x,R)} = \emptyset$, we have $E(C_2) = \emptyset$. C_1 on the other hand is part of $W_{(C_x,R)}$. Therefore, $E = \left(\bigcup_{V_i \in (C_x \cup C_1)} V_i\right) \backslash S_{(C_1,R)}$.

With these definitions in place, we prove a lemma stating that the table entries in both the probability and the utility potentials during absorption always are first degree polynomials in x.

Lemma 4.1

For any $C \in J$, if $E(C) \cap U_D = \emptyset$ then for $s \in \Omega_S$

$$\phi_S(s) = \sum_{C \backslash S} \phi'_C = a_1(s)x + a_0(s) \qquad (4.1)$$

$$\psi_S(s) = \sum_{C \backslash S} \phi'_C * \psi'_C = b_1(s)x + b_0(s), \qquad (4.2)$$

where ϕ'_C and ψ'_C denote the probability and the utility potentials of C after all its children in J have been absorbed into C.

Proof

Since the absorption of probability potentials in influence diagrams is identical to absorption in Bayesian networks, (4.1) follows immediately from Corollary 2.6. Thus, we only need to proof the result for $\psi_S(s)$.

Let $C \notin W_{(C_x,R)}$ and assume $E(C) \cap U_D = \emptyset$. Since J is a tree, we know that $\Lambda_J(C) \cap C_x = \emptyset$. Thus, none of the utility potentials absorbed into C contain x. Hereby, $\psi_S(s) = b_0(s)$, which is concordant with (4.2).

For $C \in W_{(C_x,R)}$ the lemma is shown by induction in the distance, n, between C_x and C. If $n = 0$, we have $C = C_x$, and for ψ_S we need to investigate

$$\psi_S(s) = \sum_{C_x \backslash S} \phi'_{C_x} * \psi'_{C_x} = \sum_{C_x \backslash S} (\phi_{C_x} * T(c_x)) * (\psi_{C_x} + U(c_x)),$$

where $U(c_x)$ and $T(c_x)$ originate from the absorption of the children of C_x into C_x and therefore do not contain x, neither does ψ_{C_x}. Hence, by using $\phi_{C_x}(c_x) = a_1(c_x)x + a_0(c_x)$ and Lemma 2.4,

$$\begin{aligned}\psi_S(s) &= \sum_{C_x \setminus S} ((a_1(c_x)x + a_0(c_x)) * T(c_x)) * (\psi_{C_x} + U(c_x)) \\ &= b'_1(s)x + b'_0(s)\end{aligned}$$

Now, assume the lemma is true for $n = k$, i.e.

$$\psi_{S_{(k,k+1)}}(s) = b_1(s)x + b_0(s) \tag{4.3}$$

where $S_{(k,k+1)}$ denotes the separator between C_k (the clique with distance k from C_x) and C_{k+1}, and $s \in \Omega_{S_{(k,k+1)}}$. We need to show the lemma for $n = k+1$. Therefore, with $s' \in \Omega_{S_{(k+1,k+2)}}$ we examine

$$\begin{aligned}\psi_{S_{(k+1,k+2)}}(s') &= \sum_{C_{k+1} \setminus S_{(k+1,k+2)}} \psi'_{C_{k+1}} * \psi'_{C_{k+1}} \\ &= \sum_{C_{k+1} \setminus S_{(k+1,k+2)}} \Big(\phi_{C_{k+1}} * \phi_{S_{(k,k+1)}} * T(c_{k+1})\Big) * \left(\psi_{C_{k+1}} + \frac{\psi_{S_{(k,k+1)}}}{\phi_{S_{(k,k+1)}}} + U(c_{k+1})\right) \\ &= \sum_{C_{k+1} \setminus S_{(k+1,k+2)}} T(c_{k+1}) * \Big(\phi_{C_{k+1}} * \phi_{S_{(k,k+1)}} * \psi_{C_{k+1}} \\ &\qquad + \phi_{C_{k+1}} * \phi_{S_{(k,k+1)}} * \frac{\psi_{S_{(k,k+1)}}}{\phi_{S_{(k,k+1)}}} + \phi_{C_{k+1}} * \phi_{S_{(k,k+1)}} * U(c_{k+1}))\Big)\end{aligned}$$

where $T(c_{k+1})$ and $U(c_{k+1})$ originate from absorbing probability and utility potentials of $\Lambda_j(C_{k+1}) \setminus W_{(C_x,R)}$ into C_{k+1}, respectively. Again, neither $T(c_{k+1})$, $U(C_{k+1})$ nor $\psi_{C_{k+1}}$ contain x. Note that the only clique left to be absorbed into C_{k+1} is C_k.

Before $\phi_{S_{(k,k+1)}}/\phi_{S_{(k,k+1)}}$ can be canceled out, we need to ensure the supportiveness, i.e. if $\phi_{S_{(k,k+1)}}(s) * \psi_{S_{(k,k+1)}}(s) = 0$ then $\psi_{S_{(k,k+1)}}(s) = 0$. This boils down to checking whether $\phi_{S_{(k,k+1)}}(s) = 0$ implies $\psi_{S_{(k,k+1)}}(s) = 0$. The definitions of $\psi_{S_{(k,k+1)}}$ and $\phi_{S(k,k+1)}$ ensure this. By using Corollary 2.6 and (4.3) the above reduces to

$$\begin{aligned}\psi_{S_{(k+1,k+2)}}(s') &= \sum_{C_{k+1} \setminus S_{(k+1,k+2)}} T(C_{k+1}) * \Big(\phi_{C_{k+1}} * (a_1(s)x + a_0(s)) * \psi_{C_{k+1}} \\ &\qquad + \phi_{C_{k+1}} * (b_1(s)x + b_0(s)) + \phi_{C_{k+1}} * (a_1(s)x + a_0(s)) * U(C_{k+1})) \Big),\end{aligned}$$

The three sums each give a table with elements of the form $h_1(c) + x \cdot h_2(c)$, where $c \in \Omega_{C_{k+1}}$. These sums are then combined into one table when we marginalize $C_{k+1} \setminus S_{(k+1,k+2)}$. Hence,

$$\psi_{S_{(k+1,k+2)}}(s') = b'_1(s')x + b'_0(s'),$$

This concludes the proof of the lemma.

□

Theorem 4.2

If $C_{D_2} \notin W_{(C_x,R)}$, then $\mathrm{EU}_{d \in D_1}(x) = b_1(d)x + b_0(d)$.

Proof

Transform the junction tree, J, into the equivalent junction tree, J', as illustrated in Figure 4.2.

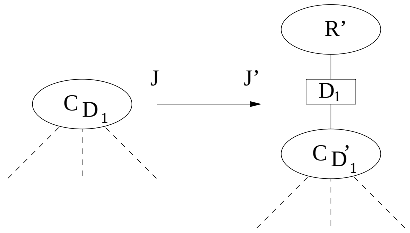


Figure 4.2: Transforming J into J', such that D_1 is placed in its own clique $R' = \{D_1\}$.

Then C'_{D_1} fulfills $E(C_{D_1}) \cap U_D = \emptyset$, because $C_{D_2} \notin W_{(C_x,R)}$. In J' we have $\mathrm{EU}_{D_1}(x) = \psi_{S_{(C'_{D_1},R)}}$. Thus, by Lemma 4.1 we get $\mathrm{EU}_{d \in D_1}(x) = b_1(d)x + b_0(d)$.

□

In the case with only one decision, we trivially have $E(C_{D_2}) \cap U_D = \emptyset$, thus, Theorem 4.2 could serve as an alternative proof of (3.3) on page 24.

We now consider the situation where $C_{D_2} \in W_{(C_x,R)}$. Assume C_{D_2} has parent C_p in J. Again, we transform the junction tree as in Figure 4.3, such that C_{D_2} receives a new clique $C' = \{S_{(C_{D_2},C_p)}, D_2\}$ as parent. Hence $S' = S_{(C_{D_2},C')} = S_{(C_{D_2},C_p)} \cup \{D_2\}$.

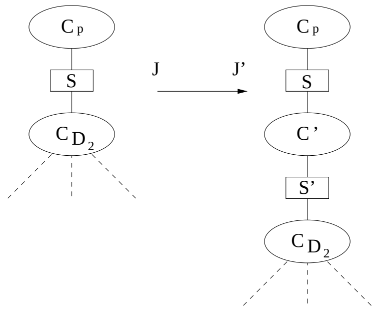


Figure 4.3: Transforming J into J', such that C_{D_2} receives a new parent C' with $C' = \{S_{(C_{D_2},C_p)}, D_2\}$ and $S' = S_{(C_{D_2},C_p)} \cup \{D_2\}$. Thus absorbing C_{D_2} into C' does not involve marginalization of D_2.

Thus, marginalizing D_2 happens on the table $\psi_{S'}$ which by using Lemma 4.1 yields

$$\max_{d\in D_2}(\psi_{S'}(s')) \quad = \quad \max_{d\in D_2}(a_1(s',d)x + a_0(s',d)). \tag{4.4}$$

For each s it is investigated whether max-marginalizing D_2 results in a so called bend point for this configuration. A **bend point** is a point, $x = c_{s'}$, where we in a decision change from one decision alternative to another. In the binary case, bend points are determined by

$$c_{s'} = \{x \mid [a_1(s',d_2) - a_1(s',\neg d_2)]x + [a_0(s',d_2) - a_0(s',\neg d_2)] = 0\} \cap]0,1[.$$

Figure 4.4 shows an example. Thus, the bend points of the potential $\psi_{s'}(s)$ in (4.4) are given by $\Gamma = \bigcup_{s'\in\Omega_{S'}} c_{s'}$.

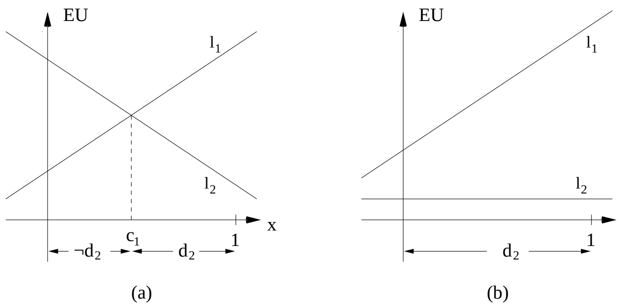


Figure 4.4: Two situation for a configuration $s' \in \Omega_{S'}$, where $l_1 = a_1(s',d_2)x + a_0(s',d_2)$ and $l_2 = a_1(s',\neg d_2)x + a_0(s',\neg d_2)$. (a) is an example of a bend point, c_1, for D_2, whereas (b) is not.

If we in some sort of symbolic propagation scheme want to use the entries of (4.4) in the calculations of future absorptions towards the root, we are forced to store up to $2|\Omega_{D_2}|$ coefficients for each configuration s. All the marginalizations of future absorptions will be sum-marginalizations. Therefore, observe that sum-marginalizing a table with bend points Γ, results in a piecewise defined function, with joint points Γ, i.e. the bend points adhere by the summation. Figure 4.5 shows the situation for a table over one binary variable, V.

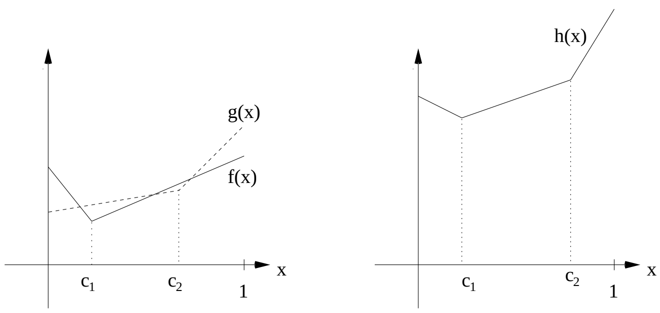


Figure 4.5: Sum-marginalization of $T(V) = (f(x), g(x))$, with $\Gamma = \{c_1, c_2\}$, i.e. $h(x) = \sum_V T(V)$. Notice how the locations of the bend points are preserved.

Multiplying probability and utility potentials of the future cliques on (4.4) will also preserve the bend points. Thus EU_{D_1} and with that $F(x)$ will have the same bend points as (4.4), i.e. Γ. This proves to be useful in our attempt to calculate r_c.

4.1.1 Calculating the Radius of Change

After analyzing EU_{D_1} in the previous section we are now able to develop an algorithm to calculate r_c in influence diagrams with two decisions and one conditional probability under study.

If Theorem 4.2 applies, the calculation of the involved coefficients and r_c follows the procedure described in Section 3.3.1.

Otherwise, our first task is to determine all the coefficients of (4.4). This is done by inserting $x = 0$ into J' and collecting to the strong root. Now, denote the utility table located at C_{D_2} after the collection by $\psi^0_{C_{D_2}}$. Similarly, the table obtained by inserting $x = 1$ and collecting is denoted $\psi^1_{C_{D_2}}$. Thus,

$$
\begin{aligned}
\psi^0_{S'}(s') &= \sum_{C_{D_2}\setminus S'} \psi^0_{C_{D_2}} = a_0(s') \\
\psi^1_{S'}(s') &= \sum_{C_{D_2}\setminus S'} \psi^1_{C_{D_2}} = a_1(s') + a_0(s')
\end{aligned}
$$

This way all the $2|\Omega_{S'}||\Omega_{D_2}|$ coefficients of the table are determined in a total of 2 evaluations.

In the binary case, our objective is to determine

$$
F(x) = [a_1(d_1) - a_1(\neg d_1)]x + [a_0(d_1) - a_0(\neg d_1)],
$$

but from the previous section we know that $F(x)$ will have all the points in Γ as bend points, i.e. $F(x)$ consists of $|\Gamma|+1$ line segments, see Figure 4.6. The coefficients of each line segment are determined as follows. Let $\Gamma' = \{c_0, c_1, \ldots, c_k, c_{k+1}\}$, where $c_0 = 0, c_{k+1} = 1$, and the remaining c_i are the bend points in Γ. Now, for all $c_i \in \Gamma'$, evaluate the influence diagram to get $y_i = F(c_i)$. Each line segment $l_i, i \in \{1, \ldots, k+1\}$ is now determined by y_{i-1} and y_i, see Figure 4.6.

Thus, a total of $2 + (|\Gamma| + 2)$ evaluations are needed in order to determine an analytical expression for $F(x)$. In a worst case situation $|\Gamma| = |\Omega_{S'}|$.

We can now apply the methods developed in Section 3.3.1 to determine r_c by considering each line segment l_i of $F(x)$. Then, for each l_i of F update the set of valid candidates by

$$
\begin{aligned}
\nabla_0 &:= \nabla_0 \cup (\nabla_0(l_i) \cap [c_{i-1}, c_i]) \\
E &:= E \cup (E(l_i) \cap [c_{i-1}, c_i]), \\
B &:= B \cup (B(l_i) \cap [c_{i-1}, c_i]),
\end{aligned}
$$

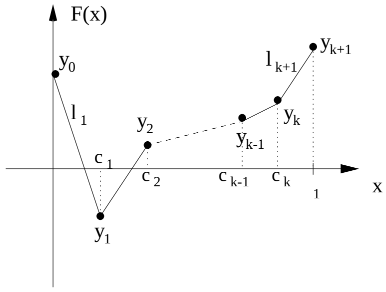


Figure 4.6: Determining $F(x)$ by analysis on each l_i, with $k = |\Gamma|$

where $\nabla_0(l_i)$, $E(l_i)$, and $B(l_i)$ denote the results of performing the analysis described in Section 3.3.2 with $F(x) = l_i$. The resulting set of points is intersected with $[c_{i-1}, c_i]$, as F and l_i only correspond within this interval. The correct value of r_c is now given by $r_c = \min \operatorname{dist}(g, \nabla_0 \cup E \cup B)$.

4.2 Two Conditional Probabilities Under Study

In this section we sketch how the method from the previous section generalizes to two conditional probabilities under study. Only the binary case is considered.

By extending Lemma 4.1 to two conditional probabilities under study, that is the functions are now of form $a_{12}x_1x_2 + a_1x_1 + a_2x_2 + a_0$, we can also give a two dimensional version of Theorem 4.2. Thus, if C_{D_2} neither has anything to do with $W_{(C_{x_1},R)}$ nor with $W_{(C_{x_2},R)}$, then r_c can be calculated by the procedure described in Section 3.3.2.

Whereas the one dimensional case had the concept of a bend point, i.e. the change from one decision alternative to another. In the two dimensional case changes between the alternatives happen along the curve of a cone intersection, as described in Section 3.3.2. This so called **bend curve**, c_i will divide $[0,1]^2$ into two areas, $A_1(s)$ and $A_2(s)$.

Using the extension of Lemma 4.1 one would in accordance with (4.4) know, that $\psi_{S'}$ is of the form

$$\max_{d \in D_2} \psi_{S'}(s', d) = \max_{d \in D_2} \left(a_{12}(s', d)x_1x_2 + a_1(s', d)x_1 + a_2(s', d)x_2 + a_0(s', d) \right). \quad (4.5)$$

Following the same principle as in the one dimensional case, a total of $2^2 = 4$ evaluations are needed to determine all the coefficients in this potential.

As with bend points, the bend curves will be preserved and accumulate into the same function as we continue to absorb cliques towards the root. Figure 4.7 shows the situation for a table over one binary variable, V.

If the potential in (4.5) has a total of $k = |\Gamma|$ bend curves, then $]0,1[^2$ for $F(x_1, x_2)$ will be divided into at most 4^k areas. Within each area, A_i, a different set of coefficients for $\{a_{12}, a_1, a_2, a_0\}$ is used. Hence, the analytical expression for $F(x)$ can be determined as

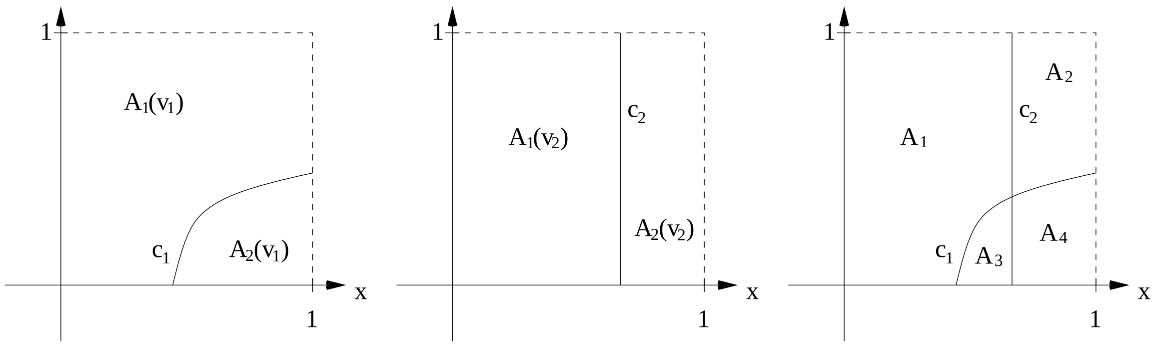


Figure 4.7: Summation of $T(V) = (v_1, v_2) = (f_{v_1}(x_1, x_2), f_{v_2}(x_1, x_2))$, with bend curves c_1 and c_2. Notice for $f'(x_1, x_2) = \sum_V T(V)$ how c_1 and c_2 divide $[0,1]^2$ into $A_1, \dots, A_4$.

follows. For each A_i find four (x_1, x_2)-points located in A_i. These points are inserted and evaluated in the influence diagram which yields four values $y_1, \dots, y_4$. These can now be used to determine the analytical expression for $F(x)$ within A_i. All together at most $4 \cdot 4^k$ propagations are needed. This number can be reduced by picking points located on the bend curves, but we abstain from performing this optimization.

All together a total of $4 + 4 \cdot 4^k$, evaluations are needed to determine $F(x)$. Notice that in a worst case situation, $k = |\Omega_{S'}|$. This number becomes quickly intractable as S' grows!

4.2.1 Calculating the Radius of Change

Once the coefficients and A_i's of $F(x)$ have been found, finding r_c is done by the following procedure. For all A_i calculate

$$r_c(A_i) \quad = \quad \min \mathrm{dist}(\bar{g}, (\nabla_0(F_{A_i}) \cup E(F_{A_i}) \cup B(F_{A_i})) \cap A_i),$$

where $\nabla_0(F_{A_i})$, $E(F_{A_i})$, and $B(F_{A_i})$ corresponds to calculating the ∇_0,E, and B sets of Section 3.3.2 with $F(x)$ equal to the one used within A_i. The minimum $r_c(A_i)$ is the value for r_c.

In this sketch, we have been somewhat sloppy with respect to possible optimization given by structure or algorithmic optimizations. This area would definitely be one of the first areas to optimize, when this work is continued.

4.3 Summary

This chapter introduced an explicit method for how to calculate r_c in influence diagrams with two decision and one parameter under study. The structure of the junction tree was investigated with Theorem 4.2 which tells when the two decision setup reduces to the one decision setup. If no reduction is possible, the expected utility as a function of x will be a continuous piecewise line, from which r_c can be found. In the case of two parameters under study, the principles are the same, but the calculations become much more extensive, and a lot of bookkeeping has to be done. A non computerized calculation of r_c is simply infeasible!

The major hurdle for such a calculation tool is the large number of propagations needed to determine the coefficients of $F(\bar{x})$, once the bend points and curves have been established. The problem is that we only extract very little information from each propagation, thus, we have a lot of overhead. Instead of inserting $\bar{x}$-values one could instead perform a new kind of propagation from S', which for each element of the potential keeps track of the involved coefficients. Unfortunately, the necessary number of coefficients is different from setup to setup, as $F(\bar{x})$ is explicitly determined by $2(|\Gamma| + 1)$ coefficients. Further research in this direction needs to be performed.

Chapter 5

Implementation

This chapter describes the implementational dimension of the theory developed in the previous sections. A tool is developed which is intended to be used by the knowledge engineer in the process of building and maintaining the knowledge base of the model. Because the tool is for sensitivity analysis purposes, it is called sat (abbreviation for Sensitivity Analysis Tool).

Since sat is merely a calculation tool without substantial user interaction or GUI, we refrain from producing an extensive and detailed analysis and design document. Instead, only the core parts of the tool are described. Thus, we present the structure of the tool and describe the different phases of the flow: The parser, the two calculation components, and the output. Special attention is given to the design considerations, expressed in Maple code, of the symbolic calculation component, whereas the other phases are described in more general terms.

5.1 The Flow and Structure of the Sensitivity Analysis Tool

The intention of sat is to function as an integrated part of a Bayesian network and influence diagram developing environment, e.g. Hugin Explorer. This allows a smooth transition between the construction and sensitivity analysis phases. Currently although, the sat-program functions only as a stand-alone unix program to be called from the shell prompt or from within another program.

The tool is programmed in C++ using the Hugin API library [Hugin Expert A/S, 1997] as well as scripts in Maple, [Kofler, 1996]. To integrate the tool within Hugin the C++ code has to be converted to the Windows platform, which is the only platform on which Hugin Explorer runs.

We will now look at the flow of the tool seen from the user's point of view and from inside the program, followed by the structure diagram of the design.

5.1.1 The Flow of sat

From the users point of view sat can be called in two different ways. One way is to embed the information about the sensitivity setup, i.e. target and study nodes, conditional probabilities etc., into the description of the model represented as a .hkb-file and then call sat with the .hkb-file as an argument. The other way is to call sat with the .hkb-file and an external script file containing the sensitivity information. This can be assembled in:

```
sat [-s <script-file>] <.hkb-file>
```

Regardless of whether the script file is given, the program has to parse either the script file or the description field of the .hkb-file. The contents of both media are identical; the grammar will be described in Section 5.2. The parser also checks for inconsistencies in the sensitivity setup and for exceedings of the limitations of the tool, e.g., with respect to the theoretic limitation on the number of decisions. Hence, before the parser runs, it must be determined whether the .hkb-file contains a Bayesian network or an influence diagram.

The parser returns a target node together with the target state and a list of the nodes under study. Each node in this list denotes the conditional probability under study by including the state under study, together with a fixed parent configuration, represented as a list of the parents of the node. Each parent node includes the state corresponding to the configuration.

The next step in the flow is the calculation component which is divided into two subcomponents, one for symbolic calculations using the theory developed in the previous sections and one for numerical analysis which serves as an indicator of the correctness of our theory and manages to cover situations not covered by our theory. Both subcomponents are divided into a component for influence diagrams and a component for Bayesian networks. Both subcomponents take the result of the parser as input an deliver their results to the output component, which manages the drawing of graphs, shows the result for the coefficients, r_c, etc. The symbolic and numeric subcomponents are described more detailed in Section 5.3 and Section 5.4 respectively.

The above description of the flow can be illustrated as in Figure 5.1. The data structures used between the components are described in the next subsection.

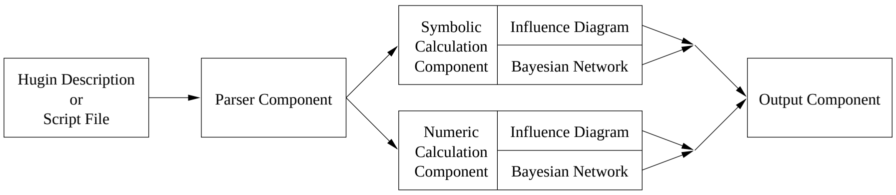


Figure 5.1: The input-output flow of sat.

5.1.2 Structural Design of sat

The structure of sat is build up around a Domain class (see Figure 5.2), which is instantiated with a .hkb file name as parameter. The class uses this parameter to open the Hugin domain (API Domain) and attach it to itself. This is followed by an examination of whether the network is a Bayesian network or an influence diagram. Finally, the Domain constructor calls the parser to retrieve the target probability or target decision (TargetNode), and the list of conditional probabilities under study, (StudyNodeList). Upon this it calls the appropriate calculation component followed by the selected method in the output component.

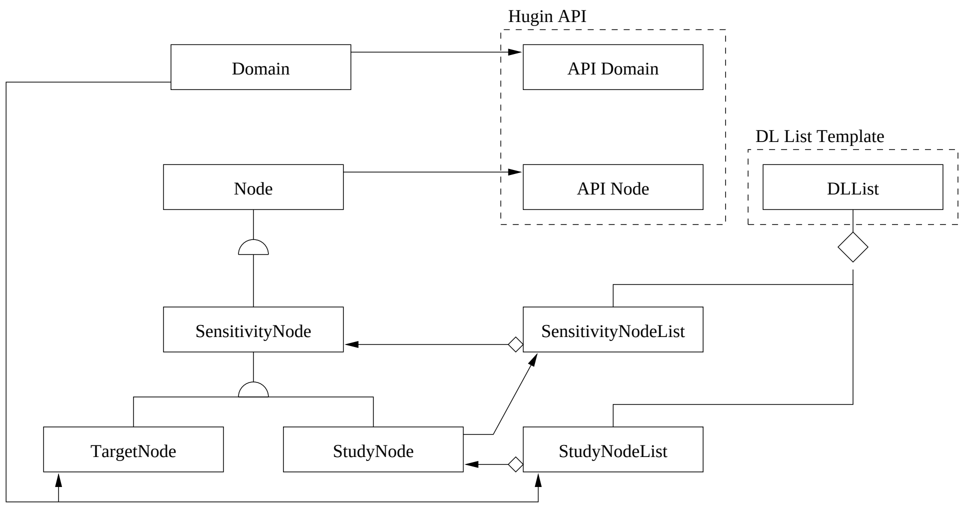


Figure 5.2: The structure diagram of sat. The boxes are classes, the half-circled lines indicate inheritance and the diamonded lines indicate inheritance from a template class, the arrows indicate a one aggregation, and a diamonded arrow indicates a "many-aggregation". Dashed lines denote external libraries

The lists in the structure diagram (SensitivityNodeList, and StudyNodeList) are all built upon the template class DLList, which is a double-linked list. The double-linked list class is further described in [George *et al.*, 1997, Appendix A.4], only minor details have been changed.

The Node class is simply an encapsulation of a node in the Hugin API. SensitivityNode is a node type used in sensitivity analysis. It contains information about which state of the variable is of interest in the sensitivity analysis. SensitivityNodeList is a list of SensitivityNode objects.

The TargetNode and StudyNode classes are inheritances of the SensitivityNode class. Though, TargetNode is just a SensitivityNode, but the StudyNode also contains a list of parents, namely a list of SensitivityNode objects. Hence, a StudyNode knows the configuration of all its parents. This is needed to specify a conditional probability of the form $P(\nu_s^i|\pi')$.

The information sent between the components in Figure 5.1 is as follows. The data structures sent to the parser is a text stream, either from the script file or from the description field in the .hkb-file. The parser returns a TargetNode object and a StudyNodeList object, both described in the previous subsection.

Each calculation component creates an appropriate specialization of a SatOutput object, which is initialized with the output of the analysis, e.g. a and b coefficients in the symbolic analysis of a Bayesian network or a large array of values in the numerical case. The specialization-structure of the output component is depicted in Figure 5.3.

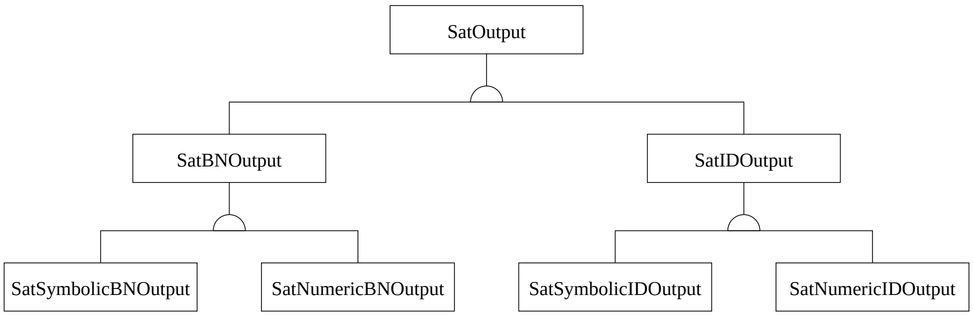


Figure 5.3: The structure diagram of the output component.

The classe SatOutput contains virtual methods to display the coefficients on the screen, to generate a Maple-file with all the relevant coefficients for further analysis, or to simply generate a visualizing graph. The leaf specialization (e.g. SatSymbolicBNOutput) then provides the appropriate implementation of these methods.

5.2 The Parser Component

The purpose of the parser component is to take some text stream, either as the description field in the `.hkb`-file or as a script file, and return the target conditional probability or target decision and also the conditional probabilities under study. The input must contain information about the target node (and its state, for Bayesian networks), a list of nodes under study given their parents, a list of evidence, and finally options to indicate the kind of output desired. The grammar to describe the sensitivity setup is independent of input-media. The grammar is as follows and an example is depicted in Figure 5.4.

body	$\rightarrow$	target study evidence options end
target	$\rightarrow$	`[target]` simple_prob
study	$\rightarrow$	`[study]` parent_prob_list
evidence	$\rightarrow$	`[evidence]` evidence_list $\mid \varepsilon$
options	$\rightarrow$	`[options]` option_type option_output
end	$\rightarrow$	`[end]`
simple_prob	$\rightarrow$	prob_item $\mid$ simple_prob_item
parent_prob_list	$\rightarrow$	parent_prob_list , parent_prob_item $\mid$ parent_prob_item
evidence_list	$\rightarrow$	evidence_list , prob_item $\mid$ prob_item
prob_item	$\rightarrow$	`P(`inner_prob`)`
simple_prob_item	$\rightarrow$	`P(`simple_inner_prob`)`
parent_prob_item	$\rightarrow$	`P(`inner_prob `|` inner_prob_list`)`
inner_prob_list	$\rightarrow$	inner_prob_list , inner_prob $\mid$ inner_prob
inner_prob	$\rightarrow$	var = state
simple_inner_prob	$\rightarrow$	var
option_type	$\rightarrow$	`TYPE` = option_type_id
option_output	$\rightarrow$	`OUTPUT` = option_output_id
option_type_id	$\rightarrow$	`symbolic` $\mid$ `numeric` $\mid$ `numeric` precision
option_output_id	$\rightarrow$	`coefficients` $\mid$ `maplefile` $\mid$ `mapleplot`

where var and state are the Hugin API names of the variables and states in our sensitivity setup. The literal precision is a numerical value indicating the precision of the numeric method, i.e. the number of propagations pr. conditional probability under study. Note, the only whitespace currently allowed is a line break after each `[...]`, after each `P(...)`, and after each option (option_type and option_output). Though, when indicating the precision of the numeric option, a single space is needed. Furthermore, parsing of evidence is not currently implemented.

Since the above grammar is pretty simple, we chose to handwrite a parser using a structure of `switch`'es embedded in a large `switch` statement, even though effective tools like Flex and Yacc exist.

As the stream is parsed, variable and state names are checked if they really exist in the underlying network. For each step, the node or state is found and the state is set and lists are updated during the parsing. After the parsing the consistency constraints with respect to the sensitivity setup are checked, e.g. a node under study cannot have the table entry for the conditional probability to be 1. Finally, the found nodes, TargetNode, and the list of nodes, StudyNodeList, are returned.

```
[TARGET]
  P(D2)
[STUDY]
  P(IMM=y|D2=y,I=n)
[EVIDENCE]
  P(D1=y)
  P(IM=y)
[OPTIONS]
  TYPE=numeric 100
  OUTPUT=mapleplot
[END]
```

Figure 5.4: An example script specifying the desired sensitivity setup for an influence diagram.

5.3 Symbolic Calculation Components

The symbolic calculation component, consists of two functions handling calculations in Bayesian networks and influence diagrams, respectively.

```
SatSymbolicBNOutput* Domain :: bnCalcCoeffs(TargetNode* targetNode, StudyNodeList* studyNodes)
SatSymbolicIDOutput* Domain :: idCalcRC(TargetNode* targetNode, StudyNodeList* studyNodes)
```

These functions are abstractions over the calculation methods described in Section 2.4 and Section 3.3, respectively. These calculations involve a lot of mathematical operations which have to be performed on the fly, since the implementation should be independent of the number of probabilities under study. Therefore we decided to use Maple to perform these calculations; a design decision which makes it hard to integrate sat within Hugin, since it is not a good marketing philosophy having a program to depend on external and expensive programming packages.

Choosing Maple allows us to concentrate on the sensitivity analysis part, otherwise a lot of resources would have had to be spent on writing and implementing algorithms, which can solve the linear equation system for the coefficients and the equation system in Laplace's multiplier method. The principle behind the cooperation of sat and Maple is to let sat generate an intermediate Maple script file, based on the number of parameters under study etc. Hereafter Maple is ran in batch-mode with this script as input. This results in the relevant output (coefficients, r_c, etc.) to be written into a temporary output file. Then, sat concludes the cooperation by reading the relevant values from the output file.

Section 5.3.1 presents the important details of how the Maple input script is generated for the Bayesian network part, whereas Section 5.3.2 merely focuses on the Maple script used to find r_c in the case of two nodes under study (also called unknowns).

5.3.1 The Bayesian Network Part

The size, m, of the `studyNodes` list determines the number of conditional probabilities under study. The 2^m values to insert into the Bayesian network for each x_i are generated by equation (2.4) on page 19. Using the Hugin API, each set of values is now inserted into the network. Denote by $O = o$ the evidence extracted by the parser. Two propagations are now performed; one with evidence $\{V_t = \nu_t, O = o\}$ and one with only $\{O = o\}$. For each propagation the probability for the evidence is read with `h_domain_get_normalization_constant`.

This results in 2^m values for $P(\nu_t, o)$ and $P(o)$, respectively. Similar to (2.13) on page 18 these values can be written as a linear equation system for the a and the b coefficients, respectively. The code below shows the generation of the a coefficient equation system.

```
ofstream maple("/tmp/sat-maple.in",ios::out);

maple << "eqnA := {" << endl;
for (i=0; i<powerSize; i++) {
  mapleWritePoly(maple,powerSize,noOfStudy,'a',x,i,numeric);
  maple << " = " << propresults[i];
  if (i !=powerSize-1) maple << "," << endl;
}
maple << "};" << endl;
```

where `mapleWritePoly` is a function that writes the correct polynomial based on the number of unknowns — e.g the numerator or seperator of (2.12) on page 18 — into the `maple` stream, and `x` is an array containing all the values to be inserted for all values of `x`. The parameter `numeric` indicates that the function should insert the values of x_i for each set of values, meaning that the value for e.g. $x[0]$ is inserted (and not the symbol `x[0]` it self, as when `symbolic` is used instead of `numeric`). The two equation systems are solved by

```
maple << "sol := solve(eqnA,{" << aunknowns << "}) union "
      <<        "solve(eqnB,{" << bunknowns << "});" << endl;
```

where e.g. `aunknowns` is a string containing all the 2^m unknowns for the a equation system, i.e. $a_0, \ldots, a_{2^m}$. The solution for all the coefficients is written into an external file by the function `mapleFprint`

```
void  mapleFprint(ofstream &maple, int powerSize,char which) {
  for (int i=0; i<powerSize; bitvec++) {
    maple << "fprintf(fd, \"%e\\n\""
          << ",subs(sol," << which << "[" << i << "]));" << endl;
  }
};
```

where `which` is either `'a'` or `'b'` depending on which coefficients we are currently writing to the ouput file. The Maple command `fprintf` is a function to print strings into a file, here indicated by the file `fd`.

5.3.2 The Influence Diagram Part

For the sake of simplicity we only consider how the generated Maple script looks in the case of two probabilities under study. Currently, it is only possible to use the generation method, if the number of decision nodes is one or if the influence diagram satisfies Theorem 4.2. The structure generalizes to m probabilities under study.

The function `idCalcRC` first extracts $\bar{g}$ from the network and determines which decision alternative, d_g, is selected with $\bar{x} = \bar{g}$. As described in Section 3.2 we now proceed our analysis for each decision alternative different from d_g. The Maple script consists of two steps; first, we need to determine the coefficients of $F(x)$, secondly, r_c is calculated as described in Section 2.4.

The generation and insertion of the 2^m values follows the form described in Section 2.4. The solution of the equation system is similar to the method described in the previous section. Therefore no details of this implementation are given here.

The second part of the Maple script calculates r_c, beginning with

```
u := vector( [x[0], x[1]] );

F := a[0] + a[1]*x[0] + a[2]*x[1] + a[3]*x[0]*x[1];
nF := grad(F, u);

dist := ( g[0] - x[0] )^2 + ( g[1] - x[1] )^2;
nDist := grad(dist, u);
```

The appropriate expressions for the $\bar{x}$ vector, $F(\bar{x})$ and $\mathrm{dist}(\bar{g}, \bar{x})$, are generated as a function of the number of probabilities under study. Hereafter expressions for $\nabla F(\bar{x})$ and $\nabla\,\mathrm{dist}(\bar{g}, \bar{x})$ are found.

```
extrema := solve( { op( map( proc(x) nF[x] = k * nDist[x] end, [1,2] ) ), F=0 },
                  { k, x[0], x[1] } );

border  := solve( { x[0]=0, F=0 }, { x[0], x[1] } ),
           solve( { x[0]=1, F=0 }, { x[0], x[1] } ),
           solve( { x[1]=0, F=0 }, { x[0], x[1] } ),
           solve( { x[0]=0, F=0 }, { x[0], x[1] } )

nabla0  := solve( { op( map( proc(x) nF[x] = 0 end, [1,2] ) ) },
                  { x[0], x[1] } );

nabla0  := op( map( FeqNull, nabla0 ) );

candidates := [ nabla0, extrema, border ];
```

The equations which determine ∇_0, E, and B are solved. The Maple command `map(p,e)` applies the procedure `p` on each operand (element) of the expression (list) `e`, and `op(e)` extracts the operands of the expression `e`. In the `extrema` case the `map` command returns a list of all coordinate functions, of which the `op` command extracts the elements, which is afterwards concatenated with `F=0`. Notice that it is important to perfom a symbolic solution (`solve`), since `extrema`, `nabla0`, and `border` all contain several solutions. The numerical solution (`fsolve`) only finds a single solution, given a start point.

The set `nabla0` is afterwards filtered with the function `FeqNull`, which throws away all solutions where $F(\bar{x}) \neq \bar{0}$, $x \in \nabla_0$. The set `extrema` contains the roots of the division of two polynomials, i.e. `extrema` could possibly contain complex solutions which need to be filtered out. Note that both `extrema` and `nabla0` only contain numeric solutions, i.e. do not depend on any x_i. On the other hand, a solution in `border` is symbolic, i.e. depending on a function in each non-fixed x_i.

The predicate function `isNotComplex` below filters out all the non-complex roots — by using the predicate function `validCandidates` to throw away all solutions not in $[0, 1]^2$. This yields a set of valid candidates, `vC`.

```
ncCand := select( isNotComplex, candidates );
vC := select( validCandidates, ncCand );
```

Next, we define a special distance function, `af`, which takes a solution element, i.e. a point, a curve, etc., where a change from one decision to another happens, and then calculating the minimum distance from $\bar{g}$ to this solution element, and returning $[\bar{x}, \text{dist}(\bar{g}, \bar{x})]$, where $\bar{x}$ is the point in the solution element with minimum distance to $\bar{g}$.

```
af := proc(y)
        [ subs( x[0] = g[0], x[1] = g[1], subs(y, x[0]) ),
          subs( x[0] = g[0], x[1] = g[1], subs(y, x[1]) ),
          subs( x[0] = g[0], x[1] = g[1], subs(y, sqrt(dist)) )]:
      end:

mCp := { op( map( af, vC) ), [ -1, -1, 1e30 ] };
```

If `y` is a point, the `dist` function can be used immediately, but if the solution `y` is given as a function in one or more x_i, some extra work has to be performed. We know that such solutions originates from `border`, i.e. it is some sort of an $(m-1)$-cube (line, plane, etc., see Figure 5.5). The point on this $(m-1)$-cube with minimum distance to $\bar{g}$ is given by the orthogonal projection of $\bar{g}$ onto the $(m-1)$-cube. As the $(m-1)$-cube is always parallel to the $(m-1)$-cube spanned by the $(m-1)$ basis vectors of the variables with freedom, the point on the $(m-1)$-cube with minimum distance to $\bar{g}$ can be found by substituting each possible freedom in `x[i]` with `g[i]`, see Figure 5.5.

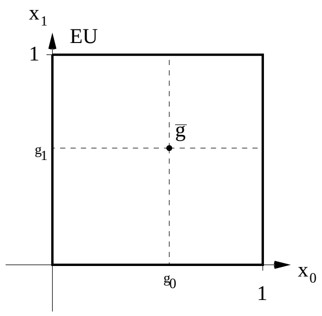


Figure 5.5: Illustration of the border in the two dimensional case. It consists of the four lines $x_0 = 0$, $x_0 = 1$, $x_1 = 0$, and $x_1 = 1$. The dashed lines show the minimum distance to each line, i.e. the variables of freedom for each line can be substituted by the corresponding values of $\bar{g}$, g_0 or g_1.

Furthermore, we add a dummy element $[-1, -1, 1 \cdot 10^{31}]$ covering the case where `mCp` is empty, i.e. $r_c = \infty$. Next, the function `minIndex` finds the index of the element in `mCp` with minimum distance.

```
rcElem := mCp[minIndex(mCp)];
```

The two first elements in `rcElem` denote the position of the change, whereas the third element contains the value of r_c. These values can now be written into the output file.

5.4 The Numerical Calculation Components

The idea of the numerical calculation components is twofold. First it serves as a verification mechanism for our symbolic calculation component and with that for the Theory developed for Bayesian networks and influence diagrams. The numeric component is simpler in structure and thus less apt to contain programming bugs and logical design flaws. Second, a numerical analysis can be performed in situations which are not covered by our theory or not implemented in the symbolic component yet. For example it is impossible for the symbolic component to calculate the radius of change for influence diagrams with two or more decision if Theorem 4.2 does not apply.

The drawback of the numerical analysis is its lack of precision and the enormous number of propagations which has to be performed, thus making a symbolic analysis both more precise and faster.

5.4.1 The Numerical Bayesian Network Calculation Component

The only available output of the numeric approach in the Bayesian network calculation component is the graph of $P(v_t|o)(\bar{x})$. Therefore the approach is limited to at most two conditional probabilities under study, since each conditional probability introduces a dimension in the graph and one dimension is used to show the value of the $P(v_t|o)(\bar{x})$.

The idea of the approach is simply to calculate the value of $P(v_t|o)(\bar{x})$ for a large number of different values for $\bar{x}$. The number of values is determined by some precision on the number of propagations. For example, if the number of propagations for one conditional probability is set to 100, then the precision is $\frac{1}{100}$ because the value interval is $[0, 1]$, for abbreviation we say that the precision is 100. This is the idea in any of the dimensions. Thus, if the precision is 100, the total number of propagations is 100^m, where m is the number of dimensions (equal to the number of conditional probabilities under study).

5.4.2 The Numerical Influence Diagram Calculation Component

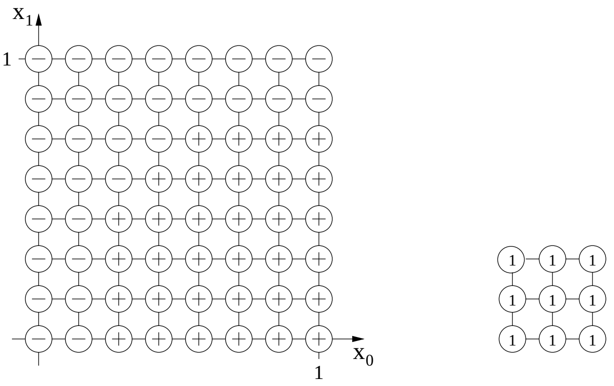
Without loss of generality, assume that the target decision node, D_t is binary. The possible outputs for the numeric approach of the influence diagram calculation component is an approximate estimation of the radius of change or a graph for $\mathrm{EU}_{D_t=d_t}$ and $\mathrm{EU}_{D_t=\neg d_t}$. The graph output again limits the amount of conditional probabilities under study to two. But, the calculation of the radius of change alone is neither limited by the number of conditional probabilities under study nor the number of decisions.

In principle the idea is similar to the one for Bayesian networks, but the realization of the method is a little harder. Here, it is also necessary to run over a number of propagations in each direction, and hereby get the expected utilities for all the decision alternatives in the target decision node. The target decision is then compared with all the other decision alternatives (one at a time), to determine whether there is an intersection. When an intersection is found, we have a candidate for the radius of change, but not necessarily the minimum.

The crucial part of this method is the comparison between the target decision and its decision alternatives. In general we are interested in finding an i such that $F(\bar{x}_i) = \mathrm{EU}_{D=d}(\bar{x}_i) - \mathrm{EU}_{D=\neg d}(\bar{x}_i) = 0$. Since this is a numerical approach, it is almost impossible to find such an i when using (fixed) discrete intervals; therefore, a sign analysis is used, i.e. the current sign of $F(\bar{x}_i)$ is compared with $F(\bar{x}_j)$ in the neighborhood, if the sign of $F(\bar{x}_i)$ is different than $F(\bar{x}_j)$, then both points are added to the list of candidates for r_c.

In the one dimensional case it is straight forward, only comparison with the last difference is necessary. If $\mathrm{sign}(F(\bar{x}_i)) \neq \mathrm{sign}(F(\bar{x}_{i-1}))$, both $\bar{x}_i$ and $\bar{x}_{i-1}$ are added to the candidate list. The precision of the method can be improved by performing some sort of interpolation between $F(\bar{x}_i)$ and $F(\bar{x}_{i-1})$, or possibly generate more data points between $\bar{x}_i$ and $\bar{x}_{i-1}$.

In the two dimensional case the comparison becomes more complicated. It is necessary to compare the sign of $F(\bar{x}_i)$ with the sign of $F(\bar{x}_j)$ for all eight neighbors $\bar{x}_j$ of $\bar{x}_i$. One way to do this is to compute a complete grid of $\mathrm{sign}(F(\bar{x}_i))$ for all $\bar{x}_i$, see Figure 5.6, and then run a mask over the grid to detect candidates for r_c. This method corresponds somewhat to edge detection in image analysis.



(a) (b)

Figure 5.6: An example of a sign grid (a) and mask (b). The idea is to detect points, where at least one of the neighbors has a different sign. The 1-mask works like a mask using the underlying sign and is ran over the entire grid. If the sum is not 9 or −9, we add the center point to the list of candidates.

Among the set of all candidates the method now calculates, which point yields the minimum distance to $\bar{g}$, and r_c is this distance. The method provides a coarse approximation of r_c, the accuracy bound is $\frac{\sqrt{k}}{100}$, if the precision is 100.

The accuracy of the method is determined by the size of the grid and whether we interpolate or generate new values once a candidate point has been found. When the work on this project is continued, optimizations with respect to accuracy and the number of propagations should be investigated. For example, could it be beneficial to use the continuity of $F(\bar{x})$ to restrict the investigation of points to the border between the plus signs and the minus signs in the sign grid. Some sort of numerical bound on the precision of the developed method should also be established.

Also a speed-compassion between the symbolic and a numeric approach should be performed. Section 4.2 indicates that a lot of overhead for bookkeeping is needed in the case of several decisions in order to determine which coefficients determine $F(\bar{x})$ where

5.5 Summary

In this chapter we have described the implementation of sat. For Bayesian networks the symbolic method is giving the desired results without limitations on the number of decisions or the number of conditional probabilities under study, $\dim(\bar{x})$. Hence, the numerical method for Bayesian networks is not very interesting because the symbolic is faster and more precise.

The speed of the symbolic Bayesian network computation could be improved by investigating whether $V_{s_i} \in \mathrm{Sen}(V_t, O)$. If this is note case, all c_M containing i are zero. Hence, the number of coefficients to be calculated are reduced.

In the influence diagram part of the symbolic calculation component we are restricted to consider only influence diagrams with at most one decision or diagrams where Theorem 4.2 applies. In theory it works for any number of conditional probabilities under study, but, in practice Maple cannot solve the equation system with more than three unknown parameters. The numerical method for influence diagrams is giving good approximations, and when considering computation time, the method sounds promising while the number of conditional probabilities under study is low, i.e. one or two.

When the project is continued three areas should be taken into consideration. First, a lot of work has to be done around the implementation, i.e. the parser must take evidence into account and handle white spaces better. Second, the numerical approach for influence diagrams should be optimized, such that the computation time is more reasonable. Third, the sketched theory for the symbolic component two decisions should be implemented.

Chapter 6

The Milk Example

In the previous chapters we have presented a theory and developed a tool for performing sensitivity analysis in influence diagrams. Therefore, we introduce an example on which we can perform sensitivity analyses using both. This example is based on an exercise found in [Jensen, 1998, Chapter 4]. The example is separated into two parts, the first part deals with the two test separately, and the second part deals with a combination of the two tests. In both parts there will be performed a sensitivity analysis on the setup.

6.1 Milk Example Episode One

In this section we first describe the problem followed by a solution using influence diagrams. Next, we perform a sensitivity analysis on the solution-network in order to see how secure the solution is given a given confidence interval on one of the parameters.

6.1.1 The Problem of the Milk Example

The general situation of the Milk Example is as follows.

> A farmer has 50 cows, each giving milk worth 5 dollars a day. Every day he delivers the milk of all the cows to the dairy in one container. The milk from a cow may be infected, and when the infected milk is mixed with the milk from all the other cows, the production from that day is infected. The dairy checks every delivery of milk, and if it is infected it is rejected, and thus worthless. This happens to the farmer twice yearly.

The test problem is described as follows.

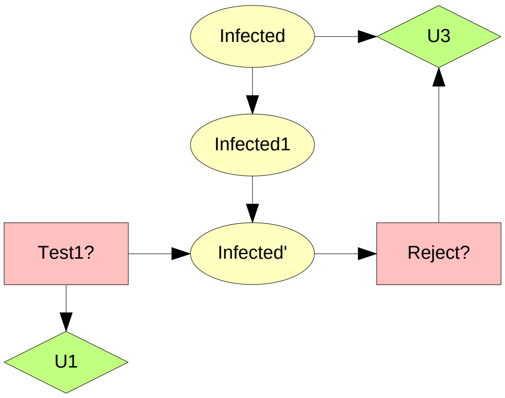
> Before milk from each cow is poured into the container the farmer can test whether it is infected. The test costs 1 cent and has no false negatives, i.e. if the milk is infected the test will always be positive; however, for non-infected milk the test gives a positive result with probability $\frac{5}{1000}$. The question now is, does it pay for the farmer to use this test?

6.1.2 A Solution of the Milk Example

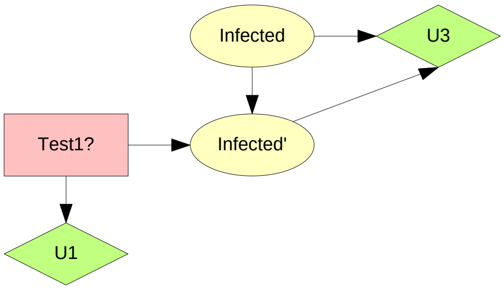
An answer to the test question could be the influence diagram given in Figure 6.1(a); for abbreviation, all lower case characters of variable names in the milk example are left out from now on, e.g. $Infected$ is turned into I.

The variable I indicates the prior probabilities of whether the milk is infected as described above. The variable $I1$ represents the outcome of the test where the false positives are taken into account. The variable I' is a dummy variable making it possible to represent the test decision, $T1?$, as well as the result of the test — if performed. This trick is described in [Jensen, 1998, Sections 4.1.3 and 4.1.4]. After $T1?$, the farmer has to decide wether to reject the milk. This is represented by $R?$.

Both utility functions of the influence diagram indicate the cost in dollars. Utility $U1$ indicates the cost of performing the test and $U3$ indicates the cost of giving the dairy infected milk and also the cost of rejecting milk. Due to no-forgetting, [Jensen, 1998, Section 4.1.2], $R?$ does not forget the decision made in $T1?$.



(a)



(b)

Figure 6.1: (a) The influence diagram for $T1?$, where we want to determine whether or not to do the test. (b) The simplified influence diagram with the same test.

The probability tables of the influence diagram in Figure 6.1 are given in Table 6.1(a). From the description of the problem, we assume that only one cow gives infected milk at a time. This seems like a valid assumption given the above description, because the probability that the infected milk is caused by two or more infected cows is very low. Then, the probability that one cow gives infected milk is $P(I = yes) = \frac{2}{365} \cdot \frac{1}{50} = \frac{1}{9125}$ because we are dealing with a union of disjoint events [DeGroot, 1985, Section 1.10].

The utility tables are given in Table 6.2. We are assuming that the farmer always gets 250 dollars, so the utilities reflect only loss. When the farmer performs a test, it has a price given by $U1$, and when the milk is infected the farmer again pays according to $U3$. The total income of the day is then, 250 dollars minus the cost of testing, the cost of rejecting, and the cost of accepting infected milk.

$P(I) =$

yes	$\frac{1}{9125}$
no	$\frac{9124}{9125}$

$P(I1|I) =$

	$I = yes$	$I = no$
$I1 = yes$	1	$\frac{5}{1000}$
$I1 = no$	0	$\frac{995}{1000}$

$P(I'|I1, T1?) =$

	$T1? = yes$	$T1? = no$
$I' = yes$	(1,0)	(0,0)
$I' = no$	(0,1)	(0,0)
$I' = unknown$	(0,0)	(1,1)

Table 6.1: The probability tables of the Milk Example. In $P(I'|I1,T1?)$ the (p_1,p_2) is given such that p_1 represents state *yes* and p_2 state *no* of $I1$.

$U1(T1?) =$

$T1? = yes$	-0.01
$T1? = no$	0

$U3(I, R?) =$

	$R? = yes$	$R? = no$
$I = yes$	-5	-250
$I = no$	-5	0

Table 6.2: The utility tables of the Milk Example.

Simplifications

To make the influence diagram in Figure 6.1(a) more suitable for our theory of sensitivity analysis calculations, some simplifications are done. The simplified solution influence diagram is pictured in Figure 6.1(b) and explained in the following.

First, the intermediate variable $I1$ can be summed out,

$$P(I'|I, T1?) = \sum_{I1} P^*(I'|I1, T1?)P(I1|I') = \sum_{I1} P(I', I1|T1?, I),$$

where P^* is the probability table from Table 6.1. Thus, the functionality of $I1$ is pushed into I' such that the probability table for $P(I'|I, T1?)$ is as in Table 6.3.

Next, the decision whether to reject the milk, $R?$, can also be integrated directly in the network. We assume that if the test says the milk is infected, we reject it (and vice versa). This can be implemented into the utility $U3$. If we did not test, we will not reject. The utility table for $U3$ is given in Table 6.4.

$$P(I'|I,T1?) = \begin{array}{|l|c|c|} \hline & T1? = yes & T? = no \\ \hline I' = yes & (1, \frac{5}{1000}) & (0,0) \\ I' = no & (0, \frac{995}{1000}) & (0,0) \\ I' = unknown & (0,0) & (1,1) \\ \hline \end{array}$$

Table 6.3: The changed probability table for the simplified influence diagram.

$$U3(I,I') = \begin{array}{|l|c|c|c|} \hline & I' = yes & I' = no & I' = unknown \\ \hline I = yes & -5 & -250 & -250 \\ I = no & -5 & 0 & 0 \\ \hline \end{array}$$

Table 6.4: The changed utility table for the simplified influence diagram, (p_1, p_2) represent state $I = yes$ and $I = no$, respectively.

Calculating the expected utility

Now, the influence diagram of the simplified Milk Example, Figure 6.1(b), and its tables are entered into Hugin. The network is compiled, and the expected utilities for the two actions can be read as

$$\text{EU}(T1?) = \begin{array}{|l|c|} \hline T1? = yes & -0.0355472 \\ T1? = no & -0.0275 \\ \hline \end{array} \tag{6.1}$$

Hereby we conclude that it does not pay for the farmer to do the test. The total expected utility for one day with a stock of 50 cows, is then

$$\underset{total}{\text{EU}} = 250 + 50 \cdot \text{MEU}(T1?) = 248.625$$

The expected utilities in (6.1) are very close, and one would now probably be curious what had happened if the test had had just a fewer false positives?

6.1.3 Sensitivity Analysis on the Milk Example

We now analyze the sensitivity with respect to one of the conditional probabilities in the Milk Example described above, corresponding to the question of how robust the conclusion is to changes in the number of false positives.

Let the decision, $T1?$, be the target decision and the conditional probability, $x = P(I' = yes|T1? = yes, I = no)$, be the conditional probability under study. Assume that we are uncertain about the value of this conditional probability, but, we know the mean value is $\mu = \frac{5}{1000}$, and assume that the confidence interval is $\pm\frac{3}{1000}$ with some certainty, say, 0.95, i.e. $P(\frac{2}{1000} \leq x \leq \frac{8}{1000}) = 0.95$.

Using the theory described in Chapter 3 and the tool described in Chapter 5 we can calculate the radius of change to be

$$r_c \; = \; 0.00160963 \tag{6.2}$$

Hence, if $P(I' = yes|T1? = yes, I = no) \leq 0.00339037$, it would pay for the farmer to take the test, otherwise not.

Equation (6.2) also gives us that the radius of change lies within the confidence interval, which is critical in the sense that the result of solving the influence diagram varies when our parameter is varied within the bounds of its confidence interval.

6.1.4 A More Reliable Test

Now, consider the following more reliable, but also more expensive, test which the farmer can perform.

> The farmer may choose another test, which cost 3 cents, it has no false negatives, and it gives false positives with a probability of $\frac{5}{10000}$. Again, the question is, does it pay for the farmer to use this test?

The influence diagram and tables are similar to the earlier ones, except for the tables $P(I''|I, T2?)$ and $U1(Test2?)$ which are shown in Tables 6.5 and 6.6, respectively. Furthermore, the nodes $I1$, I', and $T1?$ are renamed to $I2$, I'', and $T2?$, respectively.

When the values are inserted into Hugin, the expected utilities for the two actions can be extracted.

$$\mathrm{EU}(T2?) \; = \; \begin{array}{|c|c|} \hline T2? = yes & -0.0330497 \\ T2? = no & -0.0275 \\ \hline \end{array} \tag{6.3}$$

This concludes that it does not pay for the farmer to do this test, either. The total expected utility of the day with a stock of 50 cows, is then

$$\underset{total}{\mathrm{EU}} \; = \; 250 + 50 \cdot \mathrm{MEU}(T2?) \; = \; 248.625$$

As before, let $P(I'' = yes|T2? = yes, I = no)$ be the conditional probability under study. Then, the radius of change is

$$r_c \; = \; \infty,$$

from which it can be seen that no matter what probability chosen for the false positives, it would never pay for the farmer to take the test. This can also be concluded in another way, it would cost the farmer $0.03 \cdot 50 \cdot 365 \; = \; 547.5$ dollars a year to perform the test, but he only saves $2 \cdot 50 \cdot 5 \; = \; 500$ dollars, which is considerable smaller than the expenses. A sensitivity analysis gave a quick answer showing that using $T2?$ is in vain. Once the result of a sensitivity analysis is known, the knowledge engineer might want to seek explanations as done her.

$$P(I''|I, T2?) \quad = \quad \begin{array}{|l|c|c|} \hline & T2? = yes & T2? = no \\ \hline I'' = yes & (1, \frac{5}{10000}) & (0, 0) \\ I'' = no & (0, \frac{9995}{10000}) & (0, 0) \\ I'' = unknown & (0, 0) & (1, 1) \\ \hline \end{array}$$

Table 6.5: The different probability table for the simplified influence diagram for the more reliable test (compared to the first one).

$$U1(T2?) \quad = \quad \begin{array}{|l|c|} \hline T2? = yes & -0.03 \\ T2? = no & 0 \\ \hline \end{array}$$

Table 6.6: The different utility table for the simplified influence diagram for the more reliable test (compared to the first one).

6.2 Milk Example Episode Two

In the previous section the farmer had the choice between two tests, with different prices and probabilities of false positives. In this section the problem is extended as follows.

Does it pay for the farmer to combine the two tests?

6.2.1 A Solution of the Extended Milk Example

First we look at the general solution, and afterwards we do the calculations on the simplified version. Therefore, we start with extending the influence diagram from Figure 6.1 such that the two tests are incorporated by copying the structure, meaning that we introduce the decision $T2?$ and two new random variables, $I2$ and I''. The intermediate variable $I2$ indicates the probabilities concerning the false positives, and I'' is the result of $T2?$ depending on the state of $I2$.

The extended influence diagram is pictured in Figure 6.2. Since $T1?$ is the cheapest test we choose to do this decision first in the influence diagram. Next, we want to examine whether to carry out $T2?$. As before, to each test decision we assign a utility, U_1 and U_2, which indicates the cost of the tests $T1?$ and $T2?$, respectively. Again, the decision $R?$ determines whether to reject the milk. Finally, we have a utility, U_3, which describes the loss as a function of I and $R?$.

Since the probabilities and utilities are identical to the ones in the previous example, the tables are not changed compared to the ones for Figure 6.1(a).

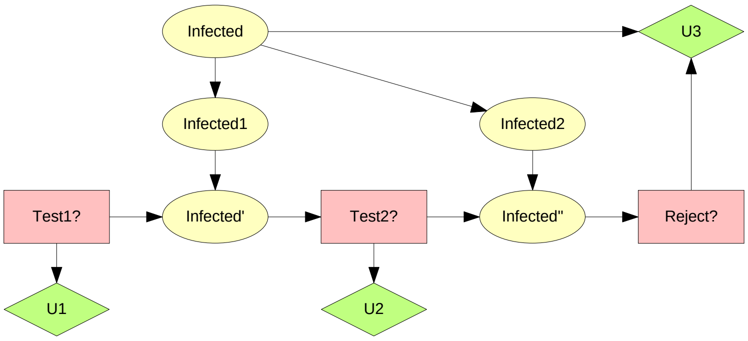


Figure 6.2: The influence diagram for $T1?$ and $T2?$, where we want to determine whether or not to do the test.

Simplifications

Once again, the influence diagram in Figure 6.2 can be simplified to make it more feasible for our theory of the sensitivity analysis. Like in the previous example, the intermediate variables, $I1$ and $I2$, are summed out, and the decision, $R?$, is included directly in the network under the assumption that if both tests are positive, we reject, otherwise if at least one of them is negative, we accept the milk. Furthermore, if we did not test, we will accept it. This functionality is integrated in $U3$. The simplified solution is pictured in Figure 6.3.

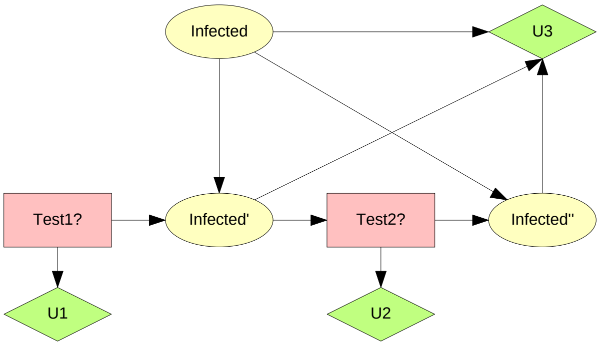


Figure 6.3: The influence diagram for $T1?$ and $T2?$, where we want to determine whether or not to combine the tests.

6.2.2 Calculating the Expected Utility

The above network and tables are entered into Hugin, from which the expected utilities of the first decision can be extracted as

$$\mathrm{EU}(T1?) \quad = \quad \begin{array}{|c|c|} \hline T1? = yes & -0.0107158 \\ T1? = no & -0.0275 \\ \hline \end{array}$$

This indicates that it pays to do $T1?$. To determine whether to perform $T2?$ depends on the result of $T1?$ and I', therefore, we calculate the expected utilities of $T2?$ given I', this can be calculated using three propagations in Hugin, one for each state in I'.

$\mathrm{EU}(T2?|I') =$

	$I' = yes$	$I' = no$	$I' = unknown$
$T2? = yes$	-0.15009	-0.04	-0.0330497
$T2? = no$	-5.01	-0.01	-0.0275

This shows that if we after $T1?$ 'know' that the milk is infected, it indeed pays to perform $T2?$. If $T1?$ says that the milk is not infected, it does not pay to perform $T2?$, because due to no false negatives we already know the milk is not infected. Finally, if we did not perform $T1$, i.e. $I' = unknown$, it does not pay to perform $T2?$, which we have already shown in the previous example in Section 6.1. Note, in this last case the expeced utilities are like the ones in (6.3) on page 59.

6.2.3 Sensitivity Analysis on the Extended Milk Example

We will now analyze the sensitivity of two different conditional probabilities (one at a time) in the extended Milk Example described above. Let again the decision $T1?$ be the target decision and the conditional probability under study be either $x = P(I' = yes|T1? = yes, I = yes)$ or $y = P(I'' = yes|T2? = yes, I = yes)$.

Sensitivity Analysis of x

Like before we assume that we are uncertain about the value of the conditional probability, x, which has mean values $\mu_x = \frac{5}{1000}$. The confidence interval is assumed to be $\pm\frac{3}{1000}$ with probability 0.95, i.e. $P(\frac{2}{1000} \leq x \leq \frac{8}{1000}) = 0.95$.

In this situation we have two decisions, therefore we need to look at the junction tree to determine whether we can calculate the radius of change symbolically or we need to use the numeric method, described in Sections 4.1.1 and 5.4.2, respectively. The junction tree for Figure 6.3 is pictured in Figure 6.4.

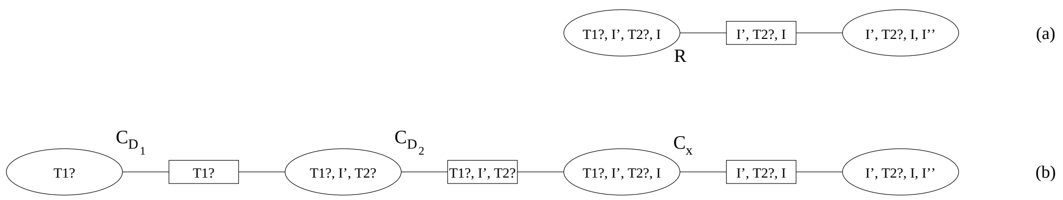
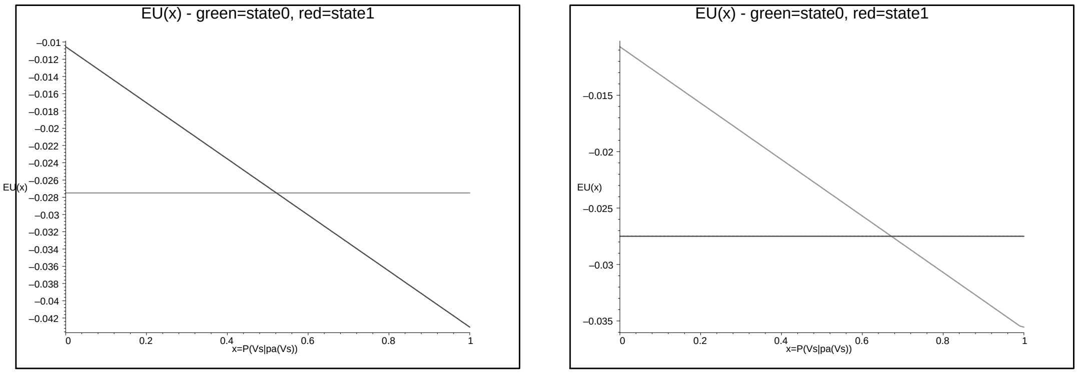


Figure 6.4: (a) The junction tree of the influence diagram in Figure 6.3. (b) The junction tree is split up such that C_{D_1}, C_{D_2} and C_x are separate cliques, corresponding to the terms in Theorem 4.2, where $C_{D_2} \in W_{(C_{x_2}, C_{D_1})}$.

From the junction tree we can conclude that the $T2?$ decision is eliminated before I', but not before $T1?$. Unfortunately, I has to be eliminated before $T2?$. Thus, the probability potential for $P(I'|I, T1?)$ can only be attached to a clique also containing $T2?$, therefore, the junction tree in Figure 6.4 cannot be transformed such that Theorem 4.2 applies. Hence, by Theorem 4.2 the symbolic calculation method of our tool cannot yet calculate the radius of change. Therefore, we need to use the numeric method, which yields for $g_x = 0.005$

$$r_c^x \approx 0.52$$

The tool, described in Chapter 5, can also give a graph illustrating the intersection, where the change of decision happens, see Figure 6.5(a).



(a) (b)

Figure 6.5: (a) The plot of EU($T1?$) for $x = P(I' = yes|T1? = yes, I = no)$. (b) The plot of EU($T1?$) for $y = P(I'' = yes|T2? = yes, I = no)$. The intersection in both plots indicates the changing point, in which the decision maker will take another decision.

Hence, if $P(I' = yes|T1? = yes, I = no) \geq 0.52$, approximately, it would no longer pay for the farmer to take the test, $T1?$. The radius of change is way larger than the confidence interval, therefore there is no critical situation here.

Sensitivity Analysis of y

Likewise, we are uncertain about the value of y, which has mean value $\mu_y = \frac{5}{10000}$. The confidence intervals are assumed to be $\pm\frac{3}{10000}$ with probability 0.95, i.e. $P(\frac{2}{10000} \leq x \leq \frac{8}{10000}) = 0.95$.

Since the variable under study, I'', is eliminated before the $T2?$ decision, which can be seen in the junction tree in Figure 6.4, then Theorem 4.2 yields that we cannot calculate the radius of change using the symbolic calculation method, because when eliminating $T2?$ a max-marginalization is introduced. Therefore, we are forced to use the numerical calculation method, for which the tool is described in Chapter 5. This yields a radius of change of

$$r_c^y \approx 0.67$$

where $g_y = 0.0005$, and the figure from the numerical approach can be seen in Figure 6.5(b).

Hence, if $P(I' = yes|T2? = yes, I = no) \geq 0.67$, approximately, it would no longer pay for the farmer to take the test, $T1?$, even when combined with $T2?$. The radius of change is much larger than the confidence interval, therefore there is no critical situation here.

6.3 Summary

We have now performed sensitivity analysis on the two parts of the milk example. In the case where the farmer only has the choice of using one of tests, none of them would be profitable for the farmer. When the farmer has the opportunity of both tests a beneficial strategy is as follows. When the first test is positive he also performs the second, and only if both tests are positive he discards the milk, otherwise he keeps it and pours it into the container.

From the sensitivity analysis on the first part of the milk example, it can be concluded that if $T1?$ had had just a fewer false positives, then, this test would pay for the farmer. But, for $T2?$ the test will never be beneficial for the farmer to perform. Sensitivity analyses yielded $r_c = \infty$ and when rationally calculating the payoff of doing the test against the profit of not rejecting non-infected milk, the payoff was much larger than the income.

When performing sensitivity analysis on the second part, it can be concluded that the model is very insensitive to the probabilities of the false positives, meaning that the probability of the false positives does not change the expected utility essentially; even if the probabilities of false positives was fifty-fifty, the farmer will do better with the tests than without them.

The sensitivity analysis performed in this chapter illustrates richly how sensitivity analysis can help checking the model, especially the conditional probabilities of the uncertain parameters. To get a better idea of how to apply sensitivity analysis in practice, one should try to investigate some larger and less fictitious diagrams.

Chapter 7

Conclusion

In this report we presented our ideas on how to provide the knowledge engineer with formal methods in the analyze phase of the construction loop of belief networks. The obtained results can be partitioned into two, theoretic results for sensitivity analysis in Bayesian networks and influence diagrams, and the sat-program as an implementational result.

On the theoretic level, we extended the work of [Coupé and van der Gaag, 1998b] such that we now are able to derive an analytic expression for the target probability, $P(v_t|o)(\bar{x})$, for any number of conditional probabilities under study. With our method, the number of propagations needed to determine the coefficients of this expression is exponential in the number of parameters under study. But, as one usually works with one or two parameters under study, this is not critical.

Hereafter, we turned our attention to influence diagrams. We claimed that a knowledge engineer is interested in knowing how much the parameters under study need to be varied before evaluating the influence diagram results in making a different decision. This information was encapsulated in the concept of the radius of change, r_c. Thus, our objective was to calculate r_c for the outermost decision in arbitrary influence diagram with an arbitrary number of parameters under study.

The presented theory only provides calculation methods for diagrams with one or two decisions. For the one decision case of calculating r_c, an arbitrary number of parameters is allowed, whereas the two decision case only covers the cases with one or two parameters under study. The $F(\bar{x})$ function which needs to be investigated was shown to become piecewise defined once the diagram contained more than one decision. The number of pieces depend on $\dim(\bar{x})$ and the quantitative part of the network. Quickly, these calculations motivate the development of an automated tool, due to the large amount of calculations and bookkeeping necessary. Again, the bottleneck seems to be the large number of propagations needed.

On the implementational level we developed a sensitivity analysis tool sat which — if better integrated — could function as an actual aid for a knowledge engineer. The tool contains a symbolic component which implements part of the above mentioned theory and a numeric component. Currently the symbolic Bayesian network part is fully implemented except for insertion of evidence. If sat should be of any use, it is certainly necessary to be able to parse evidence. The symbolic influence diagram part currently only works for diagrams with one decision, or situations that reduce to this case. Due to this and the large bookkeeping overhead a numerical solution is superior in speed for cases where $\dim(\bar{x}) = 1$ and the network contains a lot of decisions. Thus, the work on the numerical influence diagram component should be pursued further. A lot of work remains before the implementational is on the same level as the theory.

Finally, we performed an actual sensitivity analysis on a small case study concerning whether a farmer could benefit from applying a combination of two tests. The sensitivity analysis provided interesting results in a quick way, which then caused us to seek explanation in the network. When the work on this project is continued, one should definitely try to use sat in the development on some larger real world application.

7.1 Future Research

On the theoretic level the work on this project should continue on two areas. Investigation of methods to reduce the vast number of propagations needed to determine the number of pieces of $F(\bar{x})$ and generalization of the theory to any number of decisions.

In addition to this a more existentialistic problem exist. We, the authors of this report, do not have any experience in constructing belief networks modeling real world problems. This certainly weakens our statements about what methods a knowledge engineer really needs in the analyze phase of the construction loop. Our tool performs analytically well on toy examples, but the question is, are the large number of propagations needed problematic once larger networks are considered? Furthermore, once the value of r_c is known, how can it be used in the communication with the expert and to adjust the values?

We hope to be able to perform a extensive case study, where we experience the use of sat in a real world context. This requires theoretic improvements of the tool and the chance to step into the role of the knowledge engineer. First then, we can with conviction make statements about what a knowledge engineers really needs and how the users mechanistic world view can be challenged.

Appendix A

Basic Definitions, Notations, and Concepts

This appendix establishes the basic terminology used in the report. The second objective of this appendix is to serve as an introduction to graph theory and graphical models for readers not familiar with these concepts. The choice of notation for basic graph theory is presented, and an important theorem enabling efficient investigation of d-separation properties is presented. Hereafter, a shallow treatment of the comprehensive theory behind graphical models is given.

A.1 Graph Theoretic Notations and Definitions

In this section we will define some graph theoretical notations and definitions. The notations and definitions are based on [Chartrand and Oellermann, 1993], [Pearl, 1988], [Acid and de Campos, 1996], [Lauritzen *et al.*, 1990], [Lauritzen, 1996] and [Coupé and van der Gaag, 1998b].

A **graph** consists of a pair, $G = (V, E)$, where V is a finite set of **vertices**, also written $V(G)$, and E is a set of **edges**, also denoted $E(G)$. A **simple** graph is a graph without loops and multiple edges; the graphs in this report are all be simple.

An edge is a pair of vertices, $(a, b) \in E(G)$, where $a, b \in V(G)$. It can either be **directed**, meaning that if $(a, b) \in E(G)$ then $(b, a) \notin E(G)$, or **undirected**, meaning that if $(a, b) \in E(G)$ then also $(a, b) \in E(G)$. Graphically, vertices are represented by circles, undirected edges are represented by lines between circles, and directed edges by arrows between circles. Therefore, an undirected edge between a and b is written as $a \sim b$ and a directed edge from a to b as $a \to b$.

A graph is called a **directed graph** if all its edges are directed and an **undirected graph** if all its edges are undirected. In undirected graphs the edges will be represented as unordered pairs of vertices, (a, b). In this report each graph will either be directed or undirected. The **underlying graph**, $G^{\sim}$, of a directed graph, G, is the undirected graph obtained from G such that $V(G^{\sim}) = V(G)$ and if either $(a, b) \in E(G)$ or $(b, a) \in E(G)$ then both $(a, b) \in E(G^{\sim})$ and $(b, a) \in E(G^{\sim})$, i.e. substituting all the arrows by lines.

Let $A \subseteq V(G)$, then A **vertex induces** a subgraph $G_A = (A, E(G_A))$, where $E(G_A) = E(G) \cap (A \times A)$, i.e. $E(G_A)$ is obtained from G by keeping only the edges with both endpoints in A. A graph is **complete** if each vertex joins all the other vertices by either arrows or lines. The subset $A \subseteq V(G)$ is **complete** if it vertex induces a complete subgraph of G. A complete subgraph is called a **clique**, if it is a maximal complete subgraph, meaning that no other subset $A' \subseteq V(G)$, where $A \subset A'$, exists such that $G_{A'}$ is a complete vertex induced subgraph of G.

Two vertices, $a, b \in V(G)$ are called **neighbours**, if $a \sim b$ in G. The **set of neighbours** to a vertex a is denoted $\mathrm{ne}(a)$. If $a \to b$ exists, then b is called a **child** of a and a a **parent** of b. The **set of parents** of b and the **set of children** of a is denoted by $\mathrm{pa}(b)$ and $\mathrm{ch}(a)$, respectively. These definitions are extended for sets as follows.

$$\begin{aligned}
\mathrm{pa}(A) &= \bigcup_{a \in A} \mathrm{pa}(a) \backslash A \\
\mathrm{ch}(A) &= \bigcup_{a \in A} \mathrm{ch}(a) \backslash A \\
\mathrm{ne}(A) &= \bigcup_{a \in A} \mathrm{ne}(a) \backslash A
\end{aligned}$$

The **boundary**, $\mathrm{bd}(A)$, of a subset $A \subseteq V(G)$ is defined as all the vertices in $V(G)\backslash A$ that are neighbours to at least one vertex in A, that is $\mathrm{bd}(A) = \{v \in V(G)\backslash A | \exists v' \in A : v \sim v'\}$. The **inner boundary** of A is defined as $\mathrm{bd}_{\mathrm{inner}}(A) = \{v \in A | \exists v' \in \mathrm{bd}(A) : v \sim v'\}$. The **closure** of A is defined as $\mathrm{cl}(A) = A \cup \mathrm{bd}(A)$.

A **path** of length n from a vertex, $a \in V(G)$, to a vertex $b \in V(G)$ is a sequence, $a = \alpha_0, \ldots, \alpha_n = b$, of distinct vertices, such that $(\alpha_{i-1}, \alpha_i) \in E(G)$ for all $i = 1, \ldots, n$. Paths are defined for both directed and undirected graphs. If a path from a to b exists, then it is said that a **leads to** b, denoted by $a \mapsto b$. If both $a \mapsto b$ and $b \mapsto a$ then a and b **connect**, written $a \rightleftharpoons b$.

An n-**cycle** is a path of length n where $a = b$; it starts and ends at the same vertex. If the cycle contains arrows, it is called a **directed cycle**. A directed graph without cycles is called a **directed, acyclic graph (DAG)**.

In a directed graph, G, a **chain** of length n between $a \in V(G)$ and $b \in V(G)$ is said to be a sequence of distinct vertices, $a = \alpha_0, \ldots, \alpha_n = b$, such that either $\alpha_{i-1} \to \alpha_i$ or $\alpha_i \to \alpha_{i-1}$ exists.

In a directed graph, the **descendants** of a vertex $a \in V(G)$ are defined as $\mathrm{de}(a) = \{v \in V(G) | a \mapsto v\}$ and the **nondescendants** of a are defined as $\mathrm{nd}(a) = V(G)\backslash(\mathrm{de}(a) \cup \{a\})$. The ancestors of a are defined as $\mathrm{an}(a) = \{v \in V(G) | v \mapsto a\}$.

A subset $A \subseteq V(G)$ is said to be an **ancestral set**, if it contains it all its own ancestors, i.e. $\forall a \in A : \mathrm{an}(a) \cup \{a\} \subseteq A$. If A and B are ancestral sets, then $A \cap B$ is also an ancestral set, therefore the **smallest ancestral set** can be defined as $\mathrm{An}(A) = A \cup (\cup_{a \in A} \mathrm{an}(a))$.

A.2 Separation

In this section we consider separations in undirected acyclic graphs and d-separation in directed graphs. Thereafter, we give a theorem and a proof that d-separation can be reduced to separation. Finally, we give an algorithm to determine if a graph is separated.

A.2.1 Separation in Undirected Graphs

Separation of two disjoint subsets of vertices of a graph by a third subset of vertices (disjoint from the two previous sets) is defined in the following way.

Definition A.1 (Separation)
Let G be DAG, then two disjoint subsets of vertices, $A, B \subseteq V(G)$, are said to be **separated**, denoted by $\langle A|S|B\rangle_G^s$, if no path between any vertex in A and any vertex in B circumvents S, i.e. $\forall a \in A, b \in B : a \not\rightleftharpoons_{G\setminus S} b$.

A.2.2 Separation in Directed Graphs

Separation in directed, acyclic graphs is called d-spearation. Before defining d-separation, we first define a blocked chain.

Definition A.2 (Blocked chain)
Let π be a chain from a to b in a DAG, G, then π is said to be **blocked** by $S \subseteq V(G)$, if there exists a vertex $\gamma \in \pi$ such that either

- $\gamma \in S$ and no arrows of π meet head-to-head at γ, or
- $\gamma \notin S$, $\mathrm{de}(\gamma) \cap S = \emptyset$, and some arrows of π meat head-to-head at γ.

A chain is called an **active** chain, if it is not blocked by S. Then, d-separation is defined as follows.

Definition A.3 (d-separation)
Let G be a DAG and $A, B, S \subseteq V(G)$. If all chains between A and B are blocked by S, then A and B is said to be **d-separated** in G, written $\langle A|S|B\rangle_G^d$.

A.2.3 Reducing d-separation to Separation

We now present a theorem stating that d-separation, $\langle A|S|B\rangle_G^d$, reduces to separation in the moral graph of $\mathrm{An}(A \cup B \cup C)$. The theorem and proof is based on [Lauritzen *et al.*, 1990].

Theorem A.4
Let G be a DAG and A, B, and S disjoint subsets of G. Then, S d-separates A from B, if and only if S separates A and B in $(G_{\mathrm{An}(A\cup B\cup S)})^m$:

$$\langle A|S|B\rangle_G^d \quad \Leftrightarrow \quad \langle A|S|B\rangle_{(G_{\mathrm{An}(A\cup B\cup S)})^m}^s$$

Proof

This is an equivalence proof, therefore it is enough to show that if the left side does not hold, then the right does not either, and vice versa.

We first proof the forward direction by showing $\neg\langle A|S|B\rangle_G^d \Rightarrow \neg\langle A|S|B\rangle^s_{(G_{\mathrm{An}(A\cup B\cup S)})^m}$.

Suppose S does not d-separate A from B in G. Hence, there must be an active chain, π, between A and B. Without loss of generality, we can assume that the chain goes from a vertex on the inner boundary of A to a vertex of the inner boundary of B. Since A and B are not d-separated, all the vertices in π must be members of $\mathrm{An}(A \cup B \cup S)$, because if two arrows meet head-to-head in a vertex γ, then $\gamma \in S$ or $\mathrm{de}(\gamma) \cap S \neq \emptyset$, otherwise there exists a subpath away from γ that either

- meets another arrow (head-to-head), but then $\mathrm{de}(\gamma) \cap S \neq \emptyset$, or
- the subpath leads all the way to either A or B.

Since each of the head-to-head meetings causes a marriage in the moral graph, we have a path between A and B, circumventing S.

Next, we proof the backward direction, by showing $\neg\langle A|S|B\rangle^s_{(G_{\mathrm{An}(A\cup B\cup S)})^m} \Rightarrow \neg\langle A|S|B\rangle_G^d$.

Suppose A and B are not separated in $(G_{\mathrm{An}(A\cup B\cup S)})^m$. Then, there is a path in this moral graph circumventing S. Each edge in this path either corresponds to

- edges in the original graph, G, or
- marriages in the moral graph. Each marriage exists because of a head-to-head meeting at some $\gamma \in G$. Either
  - $\gamma \in S$, then $\mathrm{de}(\gamma) \cap S \neq \emptyset$, or
  - $\gamma \notin S$, then $\mathrm{de}(\gamma) \cap (A \cup B) \neq \emptyset$, because of the definition of the ancestral set, An. This means that the route of the chain has been changed, such that one less head-to-head meeting exists. By continuing this substitution of subpaths eventually leads to an active chain between A and B, in which $\neg\langle A|S|B\rangle_G^d$.

This concludes the proof.

□

A.3 Graphical Models

In this section we define the basic concepts of graphical models in the sense of Bayesian networks and influence diagrams.

A.3.1 Bayesian Networks

A Bayesian network is defined in the following way.

Definition A.5 (Bayesian network)
Let $G = (V(G), E(G))$ be a DAG with vertices $V(G) = \{V_1, \ldots, V_n\}$. Let P be a set of conditional probabilities, $P = \{P(V_i|\,\mathrm{pa}(V_i))|V_i \in V(G)\}$. A **Bayesian network** is then defined as the 2-tuple $B = (G, P)$, where G and P is defined as above.

Graphically, a Bayesian network consists of random variables as vertices in a DAG, and of arrows indicating the causal relationships (also called probabilistic dependencies) between the random variables in the network.

Each random variable has a discrete set of mutually exclusive states. The conditional probabilities indicate the probability that a variable is in a specific state given the states of its parents. Consider Figure A.1, the conditional probabilities for e.g. $P(A)$ and $P(B|A)$ are given by

$$\begin{aligned} P(A) &= (a^1, a^2) \\ P(B|A) &= \begin{array}{|c|cc|} \hline & a^1 & a^2 \\ \hline b^1 & \alpha_{11} & \alpha_{21} \\ b^2 & \alpha_{12} & \alpha_{22} \\ \hline \end{array} \end{aligned}$$

Generally, we write the set of states of a variable V_i by $\nu(V_i) = \{\nu_i^1, \ldots, \nu_i^k\}$, where V_i has k states. If a variable V_i only has two states, the variable is called a **binary variable** and the states are $\nu(V_i) = \{\nu_i, \neg\nu_i\}$. The **state space** Ω_W for $W \subseteq V(G)$ is defined as the Cartesian product $\prod_{V_i \in W} \nu(V_i)$.

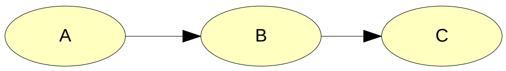


Figure A.1: An example of a Bayesian network with random variables, A, B, and C, each having two states.

The joint probability of a Bayesian network is given by the **chain rule**,

$$P(U) = \prod_{V_i \in V(G)} P(V_i|\,\mathrm{pa}(V_i)).$$

The desired probabilities can now be extracted by sum-marginalizations in $P(U)$. To make the calculations of probabilities in the Bayesian network more efficient, B is transformed into a **junction tree**. Details about the the transformation procedure and junction tree properties can be found in [Andersen *et al.*, 1997] and [Jensen, 1998]. For our investigations it is only important to know that the nodes of a junction tree — often also called cliques due to the construction process — are sets of the nodes in B, see Figure A.2. Between each pair of neighbours C_i and C_j in J the **separator** contains $C_i \cap C_j$. The transformation procedure also associates each conditional probabilities $P(V_i|\,\mathrm{pa}(V_i)$ to one of the nodes in J, the probability potential for a node is given by the multiplication of all associated tables. All separators have an initial probability table of $\underline{1}$.

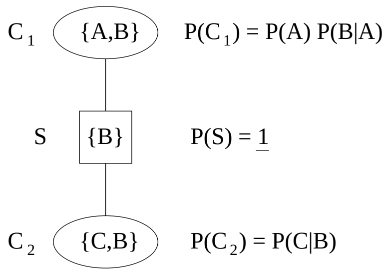


Figure A.2: The junction tree corresponding to Figure A.1. Notice how the probability potential for each node in J are given by the multiplication of all associated tables.

There now exist `collect` and `distribute` algorithms which pass the probability and evidence information in the tree by performing operations on domains smaller than U. The important operation in these algorithms is the absorption operation.

Definition A.6

Let C_i and C_j be neighbours in J with separator S, with potentials ϕ_{C_i}, ϕ_{C_j}, and ϕ_S. To **absorb** C_i into C_j means to assign the following new potentials

$$\phi'_S = \sum_{C_i \setminus S} \phi_{C_i} \qquad\qquad \phi'_{C_j} = \phi_{C_j} * \frac{\phi'_S}{\phi_S}$$

Using a sequence of these absorption operations the desired probabilities can be calculated. A call to `collect` followed by a call to `distribute` is called a **propagation**. For further details consult [Andersen *et al.*, 1997] and [Jensen, 1998].

A.3.2 Influence Diagrams

An influence diagram is a Bayesian network extended with decision variables and utility functions. Formally, an influence diagram can be defined in the following way. The definitions here are built upon [Jensen, 1998] and [Jensen *et al.*, 1994].

Definition A.7 (Influence diagram)

Let $G = (V(G), E(G))$ be a DAG. The vertices of G are represented by three disjoint sets $V(G) = V_R(G) \cup V_D(G) \cup V_U(G)$, where $V_R(G) = \{V_1, \dots, V_n\}$ is the set of random variables in G, $V_D(G) = \{D_1, \ldots, D_m\}$ is the set of decision variables, and $V_U(G) = \{\psi_1, \ldots, \psi_k\}$ is the set of utility functions.

The edges between random variables represent probabilistic dependencies. Let P be a set of conditional probabilities on the random variables, $P = \{P(V_i|\,\mathrm{pa}(V_i))|V_i \in V_R(G)\}$. The arrows into a utility indicate the domain of the utility function, i.e. $\mathrm{pa}(\psi_i) = \mathrm{dom}(\psi_i)$. Let ψ be the sum of the utility functions, $\psi = \sum_{i=1}^{k} \psi_i$. Finally, $\mathrm{pa}(D_i)$ denotes an informational dependency, that is at the time D_i is to be taken, all nodes in $\mathrm{pa}(D_i)$ are instantiated.

Now, let $U = U_R \cup U_D$, where $U_R = V_R(G)$ and $U_D = V_D(G)$, be the universe of all the variables. Then, an **influence diagram** is defined as the 3-tuple $I = (U, P, \psi)$.

Note, that our assumptions about the instantiation of $\mathrm{pa}(D_i)$ are necessary, since we want to represent actions that are under the full control of the decision-maker. The decisions must be taken in the order of their index, and there must be a directed path through all the decisions in G. An example of an influence diagram can be found in Figure A.3

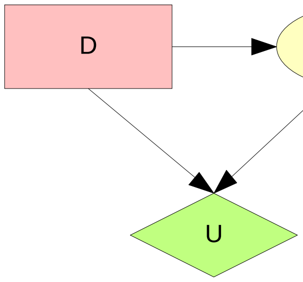


Figure A.3: An example of an influence diagram with one random variable (A), one decision node (D), and one utility function (U).

The requirements about the order of instantiation imposes a partition of $V_R(G)$ into a collection of disjoint sets, $I_0, \ldots, I_m$. For $0 < i < m$, I_i is the set of variables that is observed between D_i and D_{i+1}. Thus, I_0 are the initial evidence variables, and I_m are the variables that are never observed or observed after the last decision. This induces a partial order $\prec$ on U:

$$I_0 \prec D_1 \prec \cdots \prec D_m \prec I_m. \tag{A.1}$$

Each $V_i \in V_R(G)$ has a probability potential $\phi_{V_i} = P(V_i|\,\mathrm{pa}(V_i))$ associated. The joint probability potential is then $\phi_U = \prod_{V_i \in U_R} \phi_{V_i}$ for each possible instance of $V_D(G)$. Note, how this corresponds to the chain rule for Bayesian networks.

In influence diagrams it is an restriction that a decision cannot affect an already observed variables, i.e. the joint distribution of I_i is calculated without knowledge of $D_{i+1}, \ldots, D_m$, formally, $P(I_i|I_0, \ldots, I_{i-1}, D_1, \ldots, D_m) = P(I_i|I_0, \ldots, I_{i-1}, D_1, \ldots, D_i)$.

Taking decisions is done by following the **maximum expected utility principle**. This means that the decision alternative which maximizes the expected utility is chosen. For decision D_m the maximum expected utility is given by

$$\rho_m = \max_{D_m} \sum_{I_m} P(I_m | I_0, \ldots, I_{m-1}, D_1, \ldots, D_m) * \psi.$$

For a decision D_i with $i < m$, the maximum expected utility is $\rho_i = \max_{D_i} \sum_{I_i} P(I_i | I_0, \ldots, I_i, D_1, \ldots, D_i) * \rho_{i+1}$. Hence,

$$\rho_1 = \max_{D_1} \sum_{I_1} \max_{D_2} \sum_{I_2} \cdots \max_{D_m} \sum_{I_m} P(I_1, \ldots, I_m | I_0, D_1, \ldots, D_m) * \psi.$$

This also indicates the order, in which the variables are eliminated, corresponding to the partial order in (A.1). For more details concerning the calculations of the maximum expected utility from ρ_i consult [Jensen *et al.*, 1994].

As for Bayesian networks, we use a junction tree to achieve a more efficient way to perform the evaluation of an influence diagram. The partial order although imposes additional bounds on the construction process, see [Jensen *et al.*, 1994] for further details. The transformation of an influence diagram yields a so called strong junction tree. A **strong junction tree** is a junction tree containing at least one distinguished clique, R, called the **strong root**, such that for each pair (C_1, C_2) of adjacent cliques, where C_1 is closest to R, then an ordering of the variables in C_2 exists such that it respects $\prec$. Furthermore, the variables of the separator $C_1 \cap C_2$ precede the variables of $C_2 \backslash C_1$ in the ordering. In addition to the probability potentials, each utility potential ψ_i is by the transformation procedure associated to a clique in the strong junction tree. The utility potential of a node in J is given by the sum of all associated utility potentials.

Contrary to the passing of information in junction trees, the passing in a strong junction tree only happens through `collect` calls towards the strong root. Although `distribute` algorithms exist, it is not relevant for our application of influence diagrams. The absorption operation in strong junction trees is defined as follows.

First, we define a general marginalization operator which max- or sum-marginalizes depending on the type of the variable, i.e. marginalization of a random variable A and a decision variable D is defined as

$$\bigsqcup_{A} \phi = \sum_{A} \phi, \qquad \bigsqcup_{D} \phi = \max_{D} \phi.$$

Now, let C_1 and C_2 be adjacent cliques with separator S in J such that C_1 is closest to the root. Then, C_1 **absorbs** from C_2 by assigning the following new potentials to C_1.

$$\phi'_{C_1} = \phi_{C_1} * \phi_S, \qquad \psi'_{C_1} = \psi_{C_1} + \frac{\psi_S}{\phi_S},$$

where

$$\phi_S = \bigsqcup_{C_2 \backslash S} \phi_{C_2}, \qquad \psi_S = \bigsqcup_{C_2 \backslash S} \phi_{C_2} * \psi_{C_2}.$$

Bibliography

[Acid and de Campos, 1996] Silvia Acid and Luis M. de Campos. An algorithm for finding minimum d-separating sets in belief networks. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 3–10, San Francisco, August 1–4 1996. Morgan Kaufmann Publishers. ISBN 1-55860-412-X.

[Andersen *et al.*, 1997] Søren Andersen, Lisa Elgaard, Michael Höhle, Brian Jensen, Brian Kristiansen, and Søren Rolander Mogensen. Tænkeboks vha. bayesianske netværk og spilteori. Unpublished DAT3 project report, Fall 1997.

[Chartrand and Oellermann, 1993] Gary Chartrand and Ortrud R. Oellermann. *Applied and Algorithmic Graph Theory*. McGraw-Hill, Inc., 1993. ISBN 0-07-112575-2.

[Coupé and van der Gaag, 1998a] V. M. H. Coupé and L. C. van der Gaag. Practicable sensitivity analysis of bayesian belief networks. Technical Report UU-CS-1998-10, Utrecht University, Department of Computer Science, 1998. ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-1998/1998-10.ps.gz.

[Coupé and van der Gaag, 1998b] Veerle M. H. Coupé and Linda C. van der Gaag. Properties of sensitivity analyses of Bayesian belief networks. 1998.

[DeGroot, 1985] Morris H. DeGroot. *Probability and Statistics*. Addison-Wesley, second edition, 1985. ISBN 0-201-11366-X.

[Edwards Jr. and Penny, 1994] C. H. Edwards Jr. and David E. Penny. *Calculus with Analytic Geometry*. Prentice Hall, fourth edition, 1994. ISBN 0-13-176728-3.

[Fraleigh and Beauregard, 1990] John B. Fraleigh and Raymond A. Beauregard. *Linear Algebra*. Addison-Wesley, Reading, MA, USA, second edition, 1990. ISBN 0-201-51418-4.

[Fraleigh, 1985] John B. Fraleigh. *Calculus with Analytic Geometry*. Addison-Wesley Publishing Company, second edition, 1985. ISBN 0-210-12009-7.

[George *et al.*, 1997] Christian D. George, Jeanette F. Jensen, Jan Justensen, Brian Kristiansen, Jesper Krogh, Daniel K. Nielsen, and Morten Post. Edith — en syntaksunderstøttende editor. Unpublished DAT2 project report, Spring 1997.

[Hugin Expert A/S, 1997] Hugin Expert A/S. *Hugin API 3.1 Reference Manual*, 1997. http://www.hugin.dk/ftp/doc/api-manual.ps.

[Jensen *et al.*, 1994] Frank Jensen, Finn V. Jensen, and Søren L. Dittmer. From influence diagrams to junction trees. In R. L. de Mantaras and D. Poole, editors, *Proceedings of The Tenth Conference on Uncertainty in Artificial Intelligence*, pages 367–73. San Fransisco, CA: Morgan Kaufmann, July 1994.

[Jensen, 1998] Finn Verner Jensen. An introduction to Bayesian networks. Notes for second edtion of the book, also used in the course 'Beslutningsstøttesystemer 1', DAT3, 1998.

[Kofler, 1996] Michael Kofler. *Maple V Release 4 - Einführung und Leitfaden für den Praktiker.* Addison-Wesley (Deutschland) GmbH, 1996. ISBN 3-89319-903-9.

[Laskey, 1993] Kathryn Blackmond Laskey. Sensitivity analysis for probability assessments in bayesian networks. In David Heckerman and Abe Mamdani, editors, *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 136–142, San Mateo, CA, USA, July 1993. Morgan Kaufmann Publishers. ISBN 1-55860-258-5.

[Lauritzen *et al.*, 1990] S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H.-G. Leimer. Independence properties of markov fields. *Networks*, 20:491–505, 1990.

[Lauritzen, 1996] Stefen L. Lauritzen. *Graphical Models.* Oxford University Press, 1996. ISBN 0-19-852219-3.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan and Kaufmann, San Mateo, CA, 1988. ISBN 0-934613-73-7.

[Zhang, 1998] Nevin Lianwen Zhang. Probabilistic inference in influence diagrams. In *Proceedings of The Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 514–522. San Fransisco, CA: Morgan Kaufmann, July 1998.