
Proceedings of the Third European Workshop on

# Probabilistic Graphical Models

**Prague, Czech Republic**
**September 12–15, 2006**

**Edited by Milan Studený and Jiří Vomlel**

**Action M Agency**
**Prague, 2006**

Typeset in LaTeX.
The cover page and PGM logo: Jiří Vomlel
Graphic art: Karel Solařík

ISBN 80-86742-14-8

# Preface

These are the proceedings of the third European workshop on Probabilistic Graphical Models (PGM'06) to be held in Prague, Czech Republic, September 12-15, 2006. The aim of this series of workshops on probabilistic graphical models is to provide a discussion forum for researchers interested in this topic. The first European PGM workshop (PGM'02) was held in Cuenca, Spain in November 2002. It was a successful workshop and several of its participants expressed interest in having a biennial European workshop devoted particularly to probabilistic graphical models. The second PGM workshop (PGM'04) was held in Leiden, the Netherlands in October 2004. It was also a success; more emphasis was put on collaborative work, and the participants also discussed how to foster cooperation between European research groups.

There are two trends which can be observed in connection with PGM workshops. First, each workshop is held during an earlier month than the preceding one. Indeed, PGM'02 was held in November, PGM'04 in October and PGM'06 will be held in September. Nevertheless, I think this is a coincidence. The second trend is the increasing number of contributions (to be) presented. I would like to believe that this is not a coincidence, but an indication of increasing research interest in probabilistic graphical models.

A total of 60 papers were submitted to PGM'06 and, after the reviewing and post-reviewing phases, 41 of them were accepted for presentation at the workshop (21 talks, 20 posters) and appear in these proceedings. The authors of these papers come from 17 different countries, mainly European ones.

To handle the reviewing process (from May 15, 2006 to June 28, 2006) the PGM Program Committee was considerably extended. This made it possible that every submitted paper was reviewed by 3 independent reviewers. The reviews were handled electronically using the START conference management system.

Most of the PC members reviewed around 7 papers. Some of them also helped in the post-reviewing phase, if revisions to some of the accepted papers were desired. Therefore, I would like to express my sincere thanks here to all of the PC members for all of the work they have done towards the success of PGM'06. I think PC members helped very much to improve the quality of the contributions. Of course, my thanks are also addressed to the authors.

Further thanks are devoted to the sponsor, the DAR ÚTIA research centre, who covered the expenses related to the START system. I am also indebted to the members of the organizing committee for their help, in particular, to my co-editor, Jirka Vomlel.

My wish is for the participants in the PGM'06 workshop to appreciate both the beauty of Prague and the scientific program of the workshop. I believe there will be a fruitful discussion during the workshop and I hope that the tradition of PGM workshops will continue.

Prague, June 26, 2006

Milan Studený
PGM'06 PC Chair

# Workshop Organization

## Program Committee

**Chair:**

| | |
|---|---|
| Milan Studený | Czech Republic |

**Committee Members:**

| | |
|---|---|
| Concha Bielza | Technical University of Madrid, Spain |
| Luis M. de Campos | University of Granada, Spain |
| Francisco J. Díez | UNED Madrid, Spain |
| Marek J. Druzdzel | University of Pittsburgh, USA |
| Ad Feelders | Utrecht University, Netherlands |
| Linda van der Gaag | Utrecht University, Netherlands |
| José A. Gámez | University of Castilla La Mancha, Spain |
| Juan F. Huete | University of Granada, Spain |
| Manfred Jaeger | Aalborg University, Denmark |
| Finn V. Jensen | Aalborg University, Denmark |
| Radim Jiroušek | Academy of Sciences, Czech Republic |
| Pedro Larrañaga | University of the Basque Country, Spain |
| Peter Lucas | Radboud University Nijmegen, Netherlands |
| Fero Matúš | Academy of Sciences, Czech Republic |
| Serafín Moral | University of Granada, Spain |
| Thomas D. Nielsen | Aalborg University, Denmark |
| Kristian G. Olesen | Aalborg University, Denmark |
| José M. Peña | Linkoping University, Sweden |
| José M. Puerta | University of Castilla La Mancha, Spain |
| Silja Renooij | Utrecht University, Netherlands |
| David Ríos–Insua | Rey Juan Carlos University, Spain |
| Antonio Salmerón | University of Almeria, Spain |
| James Q. Smith | University of Warwick, UK |
| Marco Valtorta | University of South Carolina, USA |
| Jiří Vomlel | Academy of Sciences, Czech Republic |
| Marco Zaffalon | IDSIA Lugano, Switzerland |

## Additional Reviewers

| | |
|---|---|
| Alessandro Antonucci | IDSIA Lugano, Switzerland |
| Janneke Bolt | Utrecht University, Netherlands |
| Julia Flores | University of Castilla La Mancha, Spain |
| Marcel van Gerven | Radboud University Nijmegen, Netherlands |
| Yimin Huang | University of South Carolina, USA |
| Søren H. Nielsen | Aalborg University, Denmark |
| Jana Novovičová | Academy of Sciences, Czech Republic |
| Valerie Sessions | University of South Carolina, USA |
| Petr Šimeček | Academy of Sciences, Czech Republic |
| Marta Vomlelová | Charles University, Czech Republic |
| Peter de Waal | Utrecht University, Netherlands |
| Changhe Yuan | University of Pittsburgh, USA |

## Organizing Committee

**Chair:**

| | |
|---|---|
| Radim Jiroušek | Academy of Sciences, Czech Republic |

**Committee members:**

| | |
|---|---|
| Milena Zeithamlová | Action M Agency, Czech Republic |
| Marta Vomlelová | Charles University, Czech Republic |
| Radim Lněnička | Academy of Sciences, Czech Republic |
| Petr Šimeček | Academy of Sciences, Czech Republic |
| Jiří Vomlel | Academy of Sciences, Czech Republic |

# Contents

# Some Variations on the PC Algorithm

J. Abellán, M. Gómez-Olmedo, and S. Moral
Department of Computer Science and Artificial Intelligence
University of Granada
18071 - Granada, Spain

**Abstract**

This paper proposes some possible modifications on the PC basic learning algorithm and makes some experiments to study their behaviour. The variations are: to determine minimum size cut sets between two nodes to study the deletion of a link, to make statistical decisions taking into account a Bayesian score instead of a classical Chi-square test, to study the refinement of the learned network by a greedy optimization of a Bayesian score, and to solve link ambiguities taking into account a measure of their strength. It will be shown that some of these modifications can improve PC performance, depending of the objective of the learning task: discovering the causal structure or approximating the joint probability distribution for the problem variables.

## 1 Introduction

There are two main approaches to learning Bayesian networks from data. One is based on scoring and searching (Cooper and Herskovits, 1992; Heckerman, 1995; Buntine, 1991). Its main idea is to define a global measure (score) which evaluates a given Bayesian network model as a function of the data. The problem is solved by searching in the space of possible Bayesian network models trying to find the network with optimal score. The other approach (constraint learning) is based on carrying out several independence tests on the database and building a Bayesian network in agreement with tests results. The main example of this approach is PC algorithm (Spirtes et al., 1993). It can be applied to any source providing information about whether a given conditional independence relation is verified.

In the past years, searching and scoring procedures have received more attention, due to some clear advantages (Heckerman et al., 1999). One is that constraint based learning makes categorical decisions from the very beginning. These decisions are based on statistical tests that may be erroneous and these errors will affect all the future algorithm bahaviour. Another one is that scoring and search procedures allow to compare very different models by a score that can be be interpreted as the probability of being the true model. As a consequence, we can also follow a Bayesian approach considering several alternative models, each one of them with its corresponding probability, and using them to determine posterior decisions (model averaging). Finally, in score and searching approaches different combinatorial optimization techniques (de Campos et al., 2002; Blanco et al., 2003) can be applied to maximize the evaluation of the learned network. On the other hand, the PC algorithm has some advantages. One of them is that it has an intuitive basis and under some ideal conditions it has guarantee of recovering a graph equivalent to the one being a true model for the data. It can be considered as an smart selection and ordering of the questions that have to be done in order to recover a causal structure.

The basic point of this paper is that PC algorithm provides a set of strategies that can be combined with other ideas to produce good learning algorithms which can be adapted to different situations. An example of this is when van Dijk et al. (2003) propose a combination of order 0 and 1 tests of PC algorithm with an scoring and searching procedure. Here, we propose

several variations about the original PC algorithm. The first one will be a generalization of the necessary path condition (Steck and Tresp, 1999); the second will be to change the statistical tests for independence by considering decisions based on a Bayesian score; the third will be to allow the possibility of refining the network learned with PC by applying a greedy optimization of a Bayesian score; and finally the last proposal will be to delete edges from triangles in the graph following an order given by a Bayesian score (removing weaker edges first). We will show the intuitive basis for all of them and we will make some experiments showing their performance when learning Alarm network (Beinlich et al., 1989). The quality of the learned networks will be measured by the number or missing-added links and the Kullback-Leibler distance of the probability distribution associated to the learned network to the original one.

The paper is organized as follows: Section 2 is devoted to describe the fundamentals of PC algorithm; Section 3 introduces the four variations of PC algorithm; in Section 4 the results of the experiments are reported and discussed; Section 5 is devoted to the conclusions.

## 2 The PC Algorithm

Assume that we have a set of variables $\mathbf{X} = (X_1, \ldots, X_n)$ with a global probability distribution about them $P$. By an uppercase bold letter $\mathbf{A}$ we will represent a subset of variables of $\mathbf{X}$. By $I(\mathbf{A}, \mathbf{B}|\mathbf{C})$ we will denote that sets $\mathbf{A}$ and $\mathbf{B}$ are conditionally independent given $\mathbf{C}$.

PC algorithm assumes *faithfulness*. This means that there is a directed acyclic graph, $G$, such that the independence relationships among the variables in $\mathbf{X}$ are exactly those represented by $G$ by means of the d-separation criterion (Pearl, 1988). PC algorithm is based on the existence of a procedure which is able of saying when $I(\mathbf{A}, \mathbf{B}|\mathbf{C})$ is verified in graph $G$. It first tries to find the skeleton (underlying undirected graph) and on a posterior step makes the orientation of the edges. Our variations will be mainly applied to the first part (determining the skeleton). So we shall describe it with some detail:

1. Start with a complete undirected graph $G'$
2. $i = 0$
3. **Repeat**
   4. **For each** $X \in \mathbf{X}$
      5. **For each** $Y \in ADJ_X$
         6. Test whether $\exists \mathbf{S} \subseteq ADJ_X - \{Y\}$ with $|\mathbf{S}| = i$ and $I(X, Y|\mathbf{S})$
         7. **If** this set exists
            8. Make $S_{XY} = \mathbf{S}$
            9. Remove $X - Y$ link from $G'$
   10. $i = i + 1$
11. **Until** $|ADJ_X| \leq i, \quad \forall X$

In this algorithm, $ADJ_X$ is the set of nodes adjacent to $X$ in graph $G'$. The basis is that if the set of independencies is faithful to a graph, then there is not a link between $X$ and $Y$, if and only if there is a subset $\mathbf{S}$ of the adjacent nodes of $X$ such that $I(X, Y|\mathbf{S})$. For each pair of variables, $S_{XY}$ will contain such a set, if it is found. This set will be used in the posterior orientation stage.

The orientation step will proceed by looking for sets of three variables $\{X, Y, Z\}$ such that edges $X - Z, Y - Z$ are in the graph by not the edge $X - Y$. Then, if $Z \notin S_{XY}$, it orients the edges from $X$ to $Z$ and from $Y$ to $Z$ creating a v-structure: $X \rightarrow Z \leftarrow Y$. Once, these orientations are done, then it tries to orient the rest of the edges following two basic principles: not to create cycles and not to create new v-structures. It is possible that the orientation of some of the edges has to be arbitrarily selected.

If the set of independencies is faithful to a graph and we have a perfect way of determining whether $I(X, Y|\mathbf{S})$, then the algorithm has guarantee of producing a graph equivalent (represents the same set of independencies) to the original one.

However, in practice none of these conditions is verified. Independencies are decided at the light of independence statistical tests based on a set of data $\mathcal{D}$. The usual way of doing these tests is by means of a chi-square test based on the cross entropy statistic measured in the sample (Spirtes et al., 1993). Statistical tests have

errors and then, even if faithfulness hypothesis is verified, it is possible that we do not recover the original graph. The number of errors of statistical tests increases when the sample is small or the cardinality of the conditioning set **S** is large (Spirtes et al., 1993, p. 116). In both cases, due to the nature of frequentist statistical tests, there is a tendency to always decide independence (Cohen, 1988). This is one reason of doing statistical tests in increasing order of the cardinality of the sets to which we are conditioning.

Apart from no recovering the original graph, we can have another effects, as the possibility of finding cycles when orienting v-structures. In our implementation, we have always avoided cycles by reversing the arrows if necessary.

## 3 The Variations

### 3.1 Necessary Path Condition

In PC algorithm it is possible that we delete the link between $X$ and $Y$ by testing the independence $I(X, Y|\mathbf{S})$, when **S** is a set containing nodes that do not appear in a path (without cycles) from $X$ to $Y$. The inclusion of these nodes is not theoretically wrong, but statistical tests make more errors when the size of the conditioning set increases, then it can be a source of problems in practice. For this reason, Steck and Tresp (1999) proposed to reduce $ADJ_X - \{Y\}$ in Step 6, by removing all the nodes that are not in a path from $X$ to $Y$. In this paper, we will go an step further by considering any subset $CUT_{X,Y}$ disconnecting $X$ and $Y$ in the graph in which the link $X - Y$ has been deleted, playing the role of $ADJ_X - \{Y\}$. Consider that in the skeleton, we want to see whether link $X - Y$ can be deleted, then we first remove it, and if the situation is the one in Figure 1, we could consider $CUT_{X,Y} = \{Z\}$. However, in the actual algorithm (even with the necessary path condition) we consider the set $ADJ_X - \{Y\}$, which is larger, and therefore with an increased possibility of error.

Our proposal is to apply PC algorithm, but by considering in step 6 a cut set of minimum size in the graph without $X - Y$ link, as

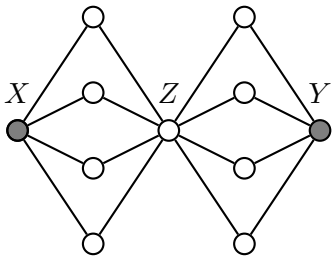



Figure 1: An small cut set

Acid and de Campos (2001) did in a different context. The computation of this set will need some extra time, but it can be done in polynomial time with a modification of Ford-Fulkerson algorithm (Acid and de Campos, 1996).

### 3.2 Bayesian Statistical Tests

PC algorithm performs a chi-square statistical test to decide about independence. However, as shown by Moral (2004), sometimes statistical tests make too many errors. They try to keep the Type I error (deciding dependence when there is independence) constant to the significance level. However, if the sample is large enough this error can be much lower by using a different decision procedure, without an important increase in Type II error (deciding independence when there is dependence). Margaritis (2003) has proposed to make statistical tests of independence for continuous variables by using a Bayesian score after discretizing them. Previously, Cooper (1997) proposed a different independence test based on a Bayesian score, but only when conditioning to 0 or 1 variable. Here we propose to do all the statistical tests by using a Bayesian Dirichlet score[1] (Heckerman, 1995) with a global sample size $s$ equal to 1.0. The test $I(X, Y|\mathbf{S})$ is carried out by comparing the scores of $X$ with **S** as parents and of $X$ with $\mathbf{S} \cup \{Y\}$ as parents. If the former is larger than the later, the variables are considered independent, and in the other case, they

[1]We have chosen this score instead of the original K2 score (Cooper and Herskovits, 1992) because this is considered more correct from a theoretical point of view (Heckerman et al., 1995).

are considered dependent. The score of $X$ with a set of parents $Pa(X) = \mathbf{Z}$ is the logarithm of:

$$\prod_{\mathbf{z}} \left( \frac{\Gamma(s')}{\Gamma(N_{\mathbf{z}} + s')} \prod_{x} \frac{\Gamma(N_{\mathbf{z},x} + s'')}{\Gamma(s'')} \right)$$

where $N_{\mathbf{z}}$ is the number of occurrences of $[\mathbf{Z} = \mathbf{z}]$ in the sample, $N_{\mathbf{z},x}$ is the number of occurrences of $[\mathbf{Z} = \mathbf{z}, X = x]$ in the sample, $s'$ is $s$ divided by the number of possible values of $\mathbf{Z}$, and $s''$ is equal to $s'$ divided by the number of values of $X$.

### 3.3 Refinement

If the statistical tests do not make errors and the faithfulness hypothesis is verified, then PC algorithm will recover a graph equivalent to the original one, but this can never be assured with finite samples. Also, even if we recover the original graph, when our objective is to approximate the joint distribution for all the variables, then depending of the sample size, it can be more convenient to use a simpler graph than the true one. Imagine that the variables follow the graph of Figure 2. This graph can be recovered by PC algorithm by doing only statistical independence tests of order 0 and 1 (conditioning to none or 1 variable). However, when we are going to estimate the parameters of the network we have to estimate a high number of probability values. This can be a too complex model (too many parameters) if the database is not large enough. In this situation, it can be reasonable to try to refine this network, taking into account the actual orientation and the size of the model. In this sense, the result of PC algorithm can be used as an starting point for a greedy search algorithm to optimize a concrete metric.

In particular, our proposal is based on the following steps:

1. Obtain an order compatible with the graph learned by PC algorithm.

2. For each node, try to delete each one of its parents or to add some of the non parents preceding nodes as parent, measuring the resulting Bayesian Dirichlet score. We make the movement with highest score difference while this is positive.

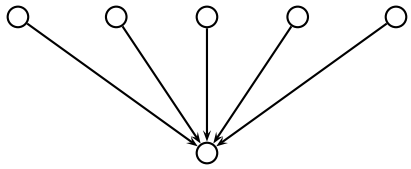


Figure 2: A too complex network.

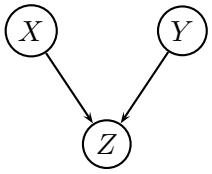



Figure 3: A simple network

Refinement can also solve some of the problems associated with the non verification of the faithfulness hypothesis. Assume for example, that we have a problem with 3 variables, $X, Y, Z$, and that the set of independencies is given by the independencies of the graph in Figure 3 plus the independence $I(Y, Z|\emptyset)$. PC algorithm will estimate a network, where the link between $Y$ and $Z$ is lost. Even if the sample is large we will estimate a too simple network which is not an I-map (Pearl, 1988). If we orient the link $X \rightarrow Z$ in PC algorithm, refinement can produce the network in Figure 3, by checking that the Bayesian score is increased (as it should be the case if $I(Z, Y|X)$ is not verified).

The idea of refining a learned Bayesian network by means of a greedy optimization of a Bayesian score has been used in a different context by Dash and Druzdzel (1999).

### 3.4 Triangles Resolution

Imagine that we have 3 variables, $X, Y, Z$, and that no independence relationship involving them is verified: each pair of variables is dependent and conditionally dependent giving the third one. As there is a tendency to decide for independence when the size of the conditioning

set is larger, then it is possible that all order 0 tests produce dependence, but when we test the independence of two variables with respect to a third one, we obtain independence. In this situation, the result of PC algorithm, will depend of the order in which tests are carried out. For example, if we ask first for the independence, $I(X, Y|Z)$, then the link $X - Y$ is deleted, but not the other two links, which will be oriented in a posterior step without creating a v-structure. If we test first $I(X, Z|Y)$, then the deleted link will be $X - Z$, but not the other two.

It seems reasonable that if one of the links is going to be removed, we should choose the weakest one. In this paper, for each 3 variables that are a triangle (the graph contains the 3 links) after order 0 tests, we measure the strength of link $X - Y$ as the Bayesian score of $X$ with $Y, Z$ as parent, minus the Bayesian score of $X$ with $Z$ as parents. For each triangle we delete the link with lowest strength (if this value is lower than 0). This is done as an intermediate step, between order 0 and order 1 conditional independence tests.

In this paper, it has been implemented only in the case in which independence tests are based on a Bayesian score, but it could be also considered in the case of Chi-square tests by considering the strength of a link equal to the p-value of the statistical independence test.

A deeper study of this type of interdependencies between the deletion of links (the presence of a link depends of the absence of other one, and vice versa) has been carried out by Steck and Tresp (1999), but the resolution of these ambiguities is not done. Hugin system (Madsen et al., 2003) allows to decide between the different possibilities by asking to the user. Our procedure could be extended to this more general setting, but at this stage the implementation has been limited to triangles, as it is, at the sample time, the most usual and simplest situation.

## 4 Experiments

We have done some experiments with the Alarm network (Beinlich et al., 1989) for testing the PC variations. In all of them, we have started with the original network and we have generated samples of different sizes by logic sampling. Then, we have tried to recover the original network from the samples by using the different variations of the PC algorithm including the orientation step. We have considered the following measures of error in this process: number of missing links, number of added links, and the Kullback-Leibler distance (Kullback, 1968) of the learned probability distribution to the original one[2]. Kullback-Leibler distance is a more appropriate measure of error when the objective is to approximate the joint probability distribution for all the variables and the measures of number of differences in links is more appropriate when our objective is to recover the causal structure of the problem. We do not consider the number of wrong orientations as our variations are mainly focused in the skeleton discovery phase of PC algorithm. The number of added or deleted links only depend of the first part of the learning algorithm (selection of the skeleton).

Experiments have been carried out in Elvira environment (Consortium, 2002), where a local computation of Kullback-Leibler distance is implemented. The different sample sizes we have used are: 100, 500, 1000, 5000, 10000, and for each sample size, we have repeated the experiment 100 times. The combinations of algorithms we have tested are the following:

Alg1 This is the algorithm with minimal separating sets, score based tests, no refinement, and triangle resolution.

Alg2 Algorithm with minimal separating sets, score based tests, refinement, and triangle resolution.

Alg3 Algorithm with adjacent nodes as separating sets, score based tests, no refinement, and triangle resolution.

Alg4 Algorithm with minimal separating sets, Chi-square tests, no refinement and no resolution of triangles.

[2]The parameters are estimated with a Bayesian Dirichlet approach with a global sample size of 2.

| | 100 | 500 | 1000 | 5000 | 10000 |
|---|---|---|---|---|---|
| Alg1 | 17.94 | 8.76 | 6.02 | 3.25 | 2.56 |
| Alg2 | 16.29 | 8.16 | 5.59 | 3.7 | 3.28 |
| Alg3 | 26.54 | 18.47 | 14.87 | 8.18 | 7.08 |
| Alg4 | 29.07 | 11.49 | 8.42 | 3.53 | 2.14 |
| Alg5 | 17.87 | 8.83 | 6.08 | 3.03 | 2.57 |

Table 1: Average number of missing links

| | 100 | 500 | 1000 | 5000 | 10000 |
|---|---|---|---|---|---|
| Alg1 | 10.42 | 4.96 | 2.87 | 2.2 | 1.98 |
| Alg2 | 26.13 | 17.52 | 16.32 | 16.01 | 15.38 |
| Alg3 | 3.06 | 0.63 | 0.14 | 0.03 | 0.01 |
| Alg4 | 14.73 | 5.96 | 5.68 | 4.78 | 4.79 |
| Alg5 | 10.46 | 4.99 | 3.21 | 1.92 | 1.9 |

Table 2: Average number of added links

Alg5 This is the algorithm with minimal separating sets, score based tests, no refinement, and no resolution of triangles.

These combinations are designed in this way, as we consider Alg1 our basic algorithm to recover the graph structure, and then we want to study the effect of the application of each one of the variations to it.

Table 1 contains the average number of missing links, Table 2 the average number of added links, Table 3 the average Kulback-Leibler distance, and finally Table 4 contains the average running times of the different algorithms. In these results we highlight the following facts:

Refinement (Alg2) increases the number of errors in the recovering of the causal structure (mainly more added links), but decreases the Kullback-Leibler distance to the original distribution. So, its application will depend of our objective: approximate the joint distribution or recover the causal structure. Refinement is fast

| | 100 | 500 | 1000 | 5000 | 10000 |
|---|---|---|---|---|---|
| Alg1 | 4.15 | 2.46 | 1.81 | 0.98 | 0.99 |
| Alg2 | 2.91 | 0.96 | 0.56 | 0.19 | 0.11 |
| Alg3 | 4.98 | 3.27 | 2.58 | 1.11 | 0.77 |
| Alg4 | 5.96 | 2.19 | 1.49 | 1.05 | 0.91 |
| Alg5 | 4.15 | 2.36 | 1.86 | 1.11 | 0.98 |

Table 3: Average Kullback-Leibler distance

| | 100 | 500 | 1000 | 5000 | 10000 |
|---|---|---|---|---|---|
| Alg1 | 2.16 | 4.1 | 5.88 | 21.98 | 42 |
| Alg2 | 2.25 | 4.12 | 5.89 | 21.98 | 42.1 |
| Alg3 | 0.33 | 1.56 | 3.34 | 20.73 | 44.14 |
| Alg4 | 2.45 | 8.9 | 13.89 | 39.42 | 68.85 |
| Alg5 | 2.21 | 4.15 | 6.02 | 22.95 | 44.4 |

Table 4: Average time

and it does not add a significant amount of extra time.

When comparing Alg1 with Alg3 (minimum size cut set vs set of adjacent nodes) we observe that with adjacent nodes fewer links are added and more ones are missing. The total amount of errors is in favour of Alg1 (minimum size cut set). This is due to the fact that Alg3 makes more conditional independence tests and with larger conditioning sets of variables, which makes more possible to delete links. Kullback-Leibler distance is better for Alg1 except for the largest sample size. A possible explanation, is that with this large sample size, we really do not miss any important link of the network, and added links can be more dangerous than deleted ones (when we delete links we are averaging distributions). With smaller sample sizes, Alg3 had a worse Kullback-Leibler as it can be missing some important links. We do not have any possible explanation to the fact that Alg1 does not improve Kullback-Leibler distance when increasing the sample size from 5000 to 10000. When comparing the time of both algorithms, we see that Alg1 needs more time (to compute minimum size cut sets) however, when the sample size is large this extra time is compensated by the lower number of statistical tests, being the total time for size 10000 lower in the case of Alg1 (with minimum size cut sets).

When comparing Alg1 and Alg4 (Score test and triangle resolution vs Chi-square tests and no triangle resolution) we observe than Alg4 always add more links and miss more links (except for the largest sample size). The total number of errors is lower for Alg1. It is meaningful the fact that the number of added links do not decrease when going from a sample of 5000 to a

sample of 10000. This is due to the fact that the probability of considering dependence when there is independence is fixed (the significance level) for large the sample sizes. So all the extra information of the larger sample is devoted to decrease the number of missing links (2.14 in Alg4 against 2.56 in Alg1), but the difference in added links is 4.79 in Alg4 against 1.98 in Alg1. So the small decreasing in missing links is at the cost of a more important error in the number of added links. Bayesian scores tests are more balanced in the two types of errors. When considering the Kullback-Leibler distance, we observe again the same situation than when comparing Alg1 and Alg2: a greater number of errors in the structure does not always imply a greater Kullback-Leibler distance. The time is always greater for Alg4.

The differences between Alg1 and Alg4 are not due to the triangles resolution in Alg1. As we will see now, triangles resolution do not really implies important changes in Alg1 performance. In fact, the effect of Chi-square tests against Bayesian tests without any other additional factor, can be seen by comparing Alg5 and Alg4. In this case, we can observe the same differences as when comparing Alg1 and Alg5.

When comparing Alg1 and Alg5 (no resolution of triangles) we see that there is not important differences in performance (errors and time) when resolution of triangles is applied. It seems that the total number of errors is decreased for intermediate sample sizes (500-1000) and there are not important differences for the other sample sizes, but more experiments are necessary. Triangles resolution do not really add a meaningful extra time. Applying this step needs some time, but the graph is simplified and posterior steps can be faster.

## 5 Conclusions

In this paper we have proposed four variations of the PC algorithm and we have tested them when learning the Alarm network. Our final recommendation would be to use the PC algorithm with score based tests, minimum size cut sets, and triangle resolution. The application of refinement step would depend of the final aim: if we want to learn the causal structure, then refinement should not be applied, but if we want to approximate the joint probability distribution, then refinement should be applied. We recognize that more extensive experiments are necessary to evaluate the application of these modifications, specially the triangle resolution. But we feel that this modification is intuitively supported, and that it could have a more important role in other situations, specially if the faithfulness hypothesis is not verified.

Other combinations could be appropriated if the objective is different, for example if we want to minimize the number of added links, then Alg3 (with adjacent nodes as cut set) could be considered.

In the future we plan to make more extensive experiments testing different networks and different combinations of these modifications. At the same time, we will consider another possible variations, as for example an algorithm mixing the skeleton and orientation steps. It is possible that some of the independencies are tested conditional to some sets, that after the orientation do not separate the two links. We also plan to study alternative scores and to study the efect of using different sample sizes. Also partial orientations can help to make the separating sets even smaller as there can be some paths which are not active without observations. This can make algorithms faster and more accurate. Finally, we think that the use of PC and its variations as starting points for greedy searching algorithms needs further research effort.

### Acknowledgments

This work has been supported by the Spanish Ministry of Science and Technology under the Algra project (TIN2004-06204-C03-02).

## References

S. Acid and L.M. de Campos. 1996. Finding minimum d-separating sets in belief networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, pages 3–10, Portland, Oregon.

S. Acid and L.M. de Campos. 2001. A hybrid methodology for learning belief networks: Benedit. *International Journal of Approximate Reasoning*, 27:235–262.

I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper. 1989. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, pages 247–256. Springer-Verlag.

R. Blanco, I. Inza, and P. Larraaga. 2003. Learning bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, 18:205–220.

W. Buntine. 1991. Theory refinement in bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann, San Francisco, CA.

J. Cohen. 1988. *Statistical power analysis for the behavioral sciences (2nd edition)*. Erlbaum, Hillsdale, NJ.

Elvira Consortium. 2002. Elvira: An environment for probabilistic graphical models. In J.A. Gmez and A. Salmern, editors, *Proceedings of the 1st European Workshop on Probabilistic Graphical Models*, pages 222–230.

G.F. Cooper and E.A. Herskovits. 1992. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

G.F. Cooper. 1997. A simple constraint-based algorithm for efficiently mining observational databases for causal relationships. *Data Mining and Knowledge Discovery*, 1:203–224.

D. Dash and M.J. Druzdzel. 1999. A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 142–149. Morgan Kaufmann.

L.M. de Campos, J.M. Fernndez-Luna, J.A. Gmez, and J. M. Puerta. 2002. Ant colony optimization for learning bayesian networks. *International Journal of Approximate Reasoning*, 31:511–549.

D. Heckerman, D. Geiger, and D.M. Chickering. 1995. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

D. Heckerman, C. Meek, and G. Cooper. 1999. A bayesian approach to causal discovery. In C. Glymour and G.F. Cooper, editors, *Computation, Causation, and Discovery*, pages 141–165. AAAI Press.

D. Heckerman. 1995. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research.

S. Kullback. 1968. *Information Theory and Statistics*. Dover, New York.

A.L. Madsen, M. Land, U.F. Kjærulff, and F. Jensen. 2003. The hugin tool for learning networks. In T.D. Nielsen and N.L. Zhang, editors, *Proceedings of ECSQARU 2003*, pages 594–605. Springer-Verlag.

D. Margaritis. 2003. *Learning Bayesian Model Structure from Data*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.

S. Moral. 2004. An empirical comparison of score measures for independence. In *Proceedings of the Tenth International Conference IPMU 2004, Vol. 2*, pages 1307–1314.

J. Pearl. 1988. *Probabilistic Reasoning with Intelligent Systems*. Morgan & Kaufman, San Mateo.

P. Spirtes, C. Glymour, and R. Scheines. 1993. *Causation, Prediction and Search*. Springer Verlag, Berlin.

H. Steck and V. Tresp. 1999. Bayesian belief networks for data mining. In *Proceedings of the 2nd Workshop on Data Mining und Data Warehousing als Grundlage moderner entscheidungsunterstuetzender Systeme*, pages 145–154.

S. van Dijk, L.C. can der Gaag, and D. Thierens. 2003. A skeleton-based approach to learning bayesian networks from data. In Nada Lavrač, Dragan Gamberger, Ljupčo Todorovski, and Hendrik Blockeel, editors, *Proceedings of the Seventh Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 132–143. Springer Verlag.

# Learning Complex Bayesian Network Features for Classification

Péter Antal, András Gézsi, Gábor Hullám and András Millinghoffer
Department of Measurement and Information Systems
Budapest University of Technology and Economics

**Abstract**

The increasing complexity of the models, the abundant electronic literature and the relative scarcity of the data make it necessary to use the Bayesian approach to complex queries based on prior knowledge and structural models. In the paper we discuss the probabilistic semantics of such statements, the computational challenges and possible solutions of Bayesian inference over complex Bayesian network features, particularly over features relevant in the conditional analysis. We introduce a special feature called Markov Blanket Graph. Next we present an application of the ordering-based Monte Carlo method over Markov Blanket Graphs and Markov Blanket sets.

In the Bayesian approach to a structural feature $F$ with values $F(G) \in \{f_i\}_{i=1}^{R}$ we are interested in the feature posterior induced by the model posterior given the observations $D_N$, where $G$ denotes the structure of the Bayesian network (BN)

$$p(f_i|D_N) = \sum_G 1(F(G) = f_i)p(G|D_N) \quad (1)$$

The importance of such inference results from (1) the frequently impractically high sample and computational complexity of the complete model, (2) a subsequent Bayesian decision-theoretic phase, (3) the availability of stochastic methods for estimating such posteriors, and (4) the focusedness of the data and the prior on certain aspects (e.g. by pattern of missing values or better understood parts of the model). Correspondingly, there is a general expectation that for small amount of data some properties of complex models can be inferred with high confidence and relatively low computation cost preserving a model-based foundation.

The irregularity of the posterior over the discrete model space of the Directed Acyclic Graphs (DAGs) poses serious challenges when such feature posteriors are to be estimated. This induced the research on the application of Markov Chain Monte Carlo (MCMC) methods for elementary features (Madigan et al., 1996; Friedman and Koller, 2000). This paper extends these results by investigating Bayesian inference about BN features with high-cardinality, relevant in classification. In Section 1 we present a unified view of BN features enriched with free-text annotations as a probabilistic knowledge base (pKB) and discuss the corresponding probabilistic semantics. In Section 2 we overview the earlier approaches to feature learning. In Section 3 we discuss structural BN features and introduce a special feature called Markov Blanket Graph or Mechanism Boundary Graph. Section 4 discusses its relevance in conditional modeling. In Section 5 we report an algorithm using ordering-based MCMC methods to perform inference over Markov Blanket Graphs and Markov Blanket sets. Section 6 presents results for the ovarian tumor domain.

## 1 BN features in pKBs

Probabilistic and causal interpretations of BN ensure that structural features can express a wide range of relevant concepts based on conditional independence statements and causal assertions (Pearl, 1988; Pearl, 2000; Spirtes et al., 2001). To enrich this approach with subjective domain knowledge via free-text annotations, we introduce the concept of Probabilistic Annotated Bayesian Network knowledge base.

**Definition 1.** A Probabilistic Annotated Bayesian Network knowledge base $K$ for a fixed set $\boldsymbol{V}$ of discrete random variables is a first-order logical knowledge base including standard graph, string and BN related predicates, relations and functions. Let $G^w$ represent a target DAG structure including all the target random variables. It includes free-text descriptions for the subgraphs and for their subsets. We assume that the models $M$ of the knowledge base vary only in $G^w$ (i.e. there is a mapping $G \leftrightarrow M$) and a distribution $p(G^w|\xi)$ is available.

A sentence $\alpha$ is any well-formed first-order formula in $K$, the probability of which is defined as the expectation of its truth

$$E_{p(\mathcal{M}|K)}[\alpha^{\mathcal{M}}] = \sum_{G} \alpha^{\mathcal{M}(G)} p(G|K).$$

where $\alpha^{\mathcal{M}(G)}$ denotes its truth-value in the model $\mathcal{M}(G)$. This hybrid approach defines a distribution over models by combining a logical knowledge base with a probabilistic model. The logical knowledge base describes the certain knowledge in the domain defining a set of models (legal worlds) and the probabilistic part ($p(G^w|\xi)$) expresses the uncertain knowledge over these worlds.

## 2 Earlier works

To avoid the statistical and computational burden of identifying complete models, related local algorithms for identifying causal relations were reported in (Silverstein et al., 2000) and in (Glymour and Cooper, 1999; Mani and Cooper, 2001). The majority of feature learning algorithms targets the learning of relevant variables for the conditional modeling of a central variable, i.e. they target the so-called *feature subset selection* (FSS) problem (Kohavi and John, 1997). Such examples are the Markov Blanket Approximating Algorithm (Koller and Sahami, 1996) and the Incremential Association Markov Blanket algorithm (Tsamardinos and Aliferis, 2003). The subgraphs of a BN as features were targeted in (Pe'er et al., 2001). The bootstrap approach inducing confidence measures for features such as compelled edges, Markov blanket membership and pairwise precedence was investigated in (Friedman et al., 1999).

On the contrary, the Bayesian framework offers many advantages such as the normative, model-based combination of prior and data allowing unconstrained application in the small sample region. Furthermore the feature posteriors can be embedded in a probabilistic knowledge base and they can be used to induce priors for other model spaces and for a subsequent learning. In (Buntine, 1991) proposed the concept of a posterior knowledge base conditioned on a given ordering for the analysis of BN models. In (Cooper and Herskovits, 1992) discussed the general use of the posterior for BN structures to compute the posterior of arbitrary features. In (Madigan et al., 1996) proposed an MCMC scheme over the space of DAGs and orderings of the variables to approximate Bayesian inference. In (Heckermann et al., 1997) considered the application of the full Bayesian approach to causal BNs. Another MCMC scheme, the ordering-based MCMC method utilizing the ordering of the variables were reported in (Friedman and Koller, 2000). They developed and used a closed form for the order conditional posteriors of Markov blanket membership, beside the earlier closed form of the parental sets.

## 3 BN features

The prevailing interpretation of BN feature learning assumes that the feature set is significantly simpler than the complete domain model providing an overall characterization as marginals and that the number of features and their values is tractable (e.g linear or quadratic in the number of variables). Another interpretation is to identify high-scoring arbitrary subgraphs or parental sets, Markov blanket subsets and estimate their posteriors. A set of simple features means a fragmentary representation for the distribution over the complete domain model from multiple, though simplified aspects, whereas using a given complex feature means a focused representation from a single, but complex point of view. A feature $F$ is complex if

the number of its values is exponential in the number of domain variables. First we cite a central concept and a theorem about relevance for variables $V = \{X_1, \ldots, X_n\}$ (Pearl, 1988).

**Definition 2.** A set of variables $MB(X_i)$ is called the Markov Blanket of $X_i$ w.r.t. the distribution $P(V)$, if $(X_i \perp\!\!\!\perp V \setminus MB(X_i) | MB(X_i))$. A minimal Markov blanket is called a Markov boundary.

**Theorem 1.** *If a distribution $P(V)$ factorizes w.r.t DAG $G$, then*

$$\forall\, i = 1, \ldots, n : (X_i \perp\!\!\!\perp V \setminus bd(X_i) | bd(X_i, G))_P,$$

*where $bd(X_i, G)$ denotes the set of parents, children and the children's other parents for $X_i$.*

So the set $bd(X_i, G)$ is a Markov blanket of $X_i$. So we will also refer to $bd(X_i, G)$ as a Markov blanket of $X_i$ in $G$ using the notation $MB(X_i, G)$ implicitly assuming that $P$ factorizes w.r.t. $G$.

The induced, symmetric pairwise relation is the Markov Blanket Membership $MBM(X_i, X_j, G)$ w.r.t. $G$ between $X_i$ and $X_j$ (Friedman et al., 1999)

$$MBM(X_i, X_j, G) \leftrightarrow X_j \in bd(X_i, G) \qquad (2)$$

Finally, we define the Markov Blanket Graph.

**Definition 3.** A subgraph of $G$ is called the Markov Blanket Graph or Mechanism Boundary Graph $MBG(X_i, G)$ of a variable $X_i$ if it includes the nodes from $MB(X_i, G)$ and the incoming edges into $X_i$ and into its children.

It is easy to show, that the characteristic property of the MBG feature is that it completely defines the distribution $P(Y | V \setminus Y)$ by the local dependency models of $Y$ and its children in a BN model $G$, in case of point parameters $(G, \boldsymbol{\theta})$ and of parameter priors satisfying global parameter independence (Spiegelhalter and Lauritzen, 1990) and parameter modularity (Heckerman et al., 1995). This property offers two interpretations for the MBG feature. From a probabilistic point of view the $MBG(G)$ feature defines an equivalence relation over the DAGs w.r.t. $P(Y | V \setminus Y)$, but clearly the MBG feature is not a unique representative of a conditionally equivalent class of BNs. From a causal point of view, this feature uniquely represents the minimal set of mechanisms including $Y$. In short, under the conditions mentioned above, this structural feature of the causal BN domain model is necessary and sufficient to support the manual exploration and automated construction of a conditional dependency model.

There is no closed formula for the posterior $p(MBG(Y, G))$, which excludes the direct use of the MBG space in optimization or in Monte Carlo methods. However, there exist a formula for the order conditional posterior with polynomial time complexity if the size of the parental sets is bounded by $k$

$$\begin{aligned} p(MBG(Y, G) = mbg | D_N) = \\ p(pa(Y, mbg) | D_N) \\ \prod_{\substack{Y \prec X_i \\ Y \in pa(X_i, mbg)}} p(pa(X_i, mbg) | D_N) \\ \prod_{\substack{Y \prec X_i \\ Y \notin pa(X_i, mbg)}} \sum_{Y \notin pa(X_i)} p(pa(X_i) | D_N). \end{aligned} \qquad (3)$$

The cardinality of the $MBG(Y)$ space is still super-exponential (even if the number of parents is bounded by $k$). Consider an ordering of the variables such that $Y$ is the first and all the other variables are children of it, then the parental sets can be selected independently, so the number of alternatives is in the order of $(n-1)^{n^2}$ (or $(n-1)^{(k-1)(n-1)}$). However, if $Y$ is the last in the ordering, then the number of alternatives is of the order $2^{n-1}$ or $(n-1)^{(k)}$). In case of $MBG(Y, G)$, variable $X_i$ can be (1) non-occurring in the MBG, (2) a parent of $Y$ ($X_i \in pa(Y, G)$), (3) a child of $Y$ ($X_i \in ch(Y, G)$) and (4) (pure) other parent ($(X_i \notin pa(Y, G) \wedge (X_i \in pa(ch(Y)_j))))$). These types correspond to the irrelevant (1) and strongly relevant (2,3,4) categories (see, Def. 4). The number of DAG models $G(n)$ compatible with a given MBG and ordering $\prec$ can be computed as follows: the contribution of the variables $X_i \prec Y$ without any constraint and the contribution of the variables $Y \prec X_i$ that are

not children of Y, which is still $2^{\mathcal{O}((k)(n)log(n))}$ (note certain sparse graphs are compatible with many orderings).

## 4 Features in conditional modeling

In the conditional Bayesian approach the relevance of predictor variables (features in this context) can be defined in an asymptotic, algorithm-, model- and loss-free way as follows

**Definition 4.** A feature $X_i$ is strongly relevant iff there exists some $x_i, y$ and $s_i = x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ for which $p(x_i, s_i) > 0$ such that $p(y|x_i, s_i) \neq p(y|s_i)$. A feature $X_i$ is weakly relevant iff it is not strongly relevant, and there exists a subset of features $S'_i$ of $S_i$ for which there exists some $x_i, y$ and $s'_i$ for which $p(x_i, s'_i) > 0$ such that $p(y|x_i, s'_i) \neq p(y|s'_i)$. A feature is relevant if it is either weakly or strongly relevant; otherwise it is irrelevant (Kohavi and John, 1997).

In the so-called filter approach to feature selection we have to select a minimal subset $X'$ which fully determines the conditional distribution of the target ($p(Y|X) = p(Y|X')$). If the conditional modeling is not applicable and a domain model-based approach is necessary then the Markov boundary property (feature) seems to be an ideal candidate for identifying relevance. The following theorem gives a sufficient condition for uniqueness and minimality (Tsamardinos and Aliferis, 2003).

**Theorem 2.** *If the distribution $P$ is stable w.r.t. the DAG $G$, then the variables $bd(Y, G)$ form a unique and minimal Markov blanket of $Y$, $MB(Y)$. Furthermore, $X_i \in MB(Y)$ iff $X_i$ is strongly relevant.*

However, the MBG feature provides a more detailed description about relevancies. As an example, consider that a logistic regression (LR) model without interaction terms and a Naive BN model can be made conditionally equivalent using a local and transparent parameter transformation. If the distribution contains additional dependencies, then the induced conditional distribution has to be represented by a LR model with interaction terms.

## 5 Estimating complex features

The basic task is the estimation of the expectation of a given random variable over the space of DAGs with a specified confidence level in Eq. 1. We assume complete data, discrete domain variables, multinomial local conditional distributions and Dirichlet parameter priors. It ensures efficiently computable closed formulas for the (unnormalized) posteriors of DAGs. As this posterior cannot be sampled directly and the construction of an approximating distribution is frequently not feasible, the standard approach is to use MCMC methods such as the Metropolis-Hastings over the DAG space (see e.g. (Gamerman, 1997; Gelman et al., 1995)).

The DAG-based MCMC method for estimating a given expectation is generally applicable, but for certain types of features such as Markov blanket membership an improved method, the so-called ordering-based MCMC method can be applied, which utilizes closed-forms of the order conditional feature posteriors computable in $\mathcal{O}(n^{k+1})$ time, where $k$ denotes the maximum number of parents (Friedman and Koller, 2000).

In these approaches the problem is simplified to the estimation of separate posteriors. However, the number of target features can be as high as $10^4 - 10^6$ even for a given type of pairwise features and moderate domain complexity. This calls for a decision-theoretic report of selection and estimation of the features, but here we use a simplified approach targeting the selection-estimation of the $K$ most probable feature values. Because of the exponential number of feature values a search method has to be applied either iteratively or in an integrated fashion. The first approach requires the offline storage of orderings and corresponding common factors, so we investigated this latter option. The integrated feature selection-estimation is particularly relevant for the ordering-based MC methods, because it does not generate implicitly "high-scoring" features and features that are not part of the solution cause extra computational costs in estimation.

The goal of search within the MC cycle at step $l$ is the generation of MBGs with high

order conditional posterior, potentially using the already generated MBGs and the posteriors $p(MBG|\prec_l, D_N)$. To facilitate the search we define an order conditional MBG state space based on the observation that the order conditionally MAP MBG can be found in $\mathcal{O}(n^{k+1})$ time with a negligible constant increase only. An MBG state is represented by an $n'$ dimensional vector $\underline{s}$, where $n'$ is the number of variables not preceding the target variable $Y$ in the ordering $\prec$:

$$n' = \sum_{i=1}^{n} 1(Y \preceq X_i) \tag{4}$$

The range of the values are integers $s_i = 0, \ldots, r_i$ representing either separate parental sets or (in case of $X_i$ where $Y \prec_l X_i$) a special set of parental sets not including the target variable. The product of the order conditional posteriors of the represented sets of parental sets gives the order conditional posterior of the represented MBG state in Eq. 3. We ensure that the conditional posteriors of the represented sets of parental sets are monotone decreasing w.r.t. their indices:

$$\forall s_i < s_i' : p(s_i|D_N, \prec) \geq p(s_i'|D_N, \prec) \tag{5}$$

which can be constructed in $\mathcal{O}(n^{k+1} \log(\max_i r_i))$ time, where $k$ is the maximum parental set size.

This MBG space allows the application of efficient search methods. We experimented with a direct sampling, top-sampling and a deterministic search. The direct sampling was used as a baseline, because it does need the MBG space. The top-sampling method is biased towards sampling MBGs with high order conditional posterior, by sampling only from the L most probable sets of parental sets for each $Y \precsim X_i$. The uniform-cost search constructs the MBG space, then performs a uniform-cost search to a maximum number of MBGs or to threshold $p(MBG^{MAP,\prec}|\prec, D_N)/\rho^S$.

The pseudo code of searching and estimating MAP values for the MBG feature of a given variable is shown in Alg. 1 (for simplicity the estimation of simple classification features such as edge and MBM relations, and the estimation of the MB features of a given variable using the estimated MAP MBG collection is not shown).

**Algorithm 1** Search and estimation of classification features using the MBG-ordering spaces

**Require:** $p(\prec)$,p$(pa(X_i)|\prec)$,k,R,$\rho$,$L^S$,$\rho^S$,$L^T$,M;
**Ensure:** K MAP feature value with estimates
  Cache order-free parental posteriors $\Pi = \{\forall\, i, |pa(X_i)| \leq k : p(pa(X_i)|D_N)\}$
  Initialize MCMC, the MBG-tree $\mathcal{T}$, MBM and edge posterior matrices $\mathcal{R}$, $\mathcal{E}$;
  Insert a priori specified MBGs in $\mathcal{T}$;
  **for** $l = 0$ to $M$ **do** {the sampling cycle}
    Draw next ordering;
    Cache order specific common factors $\Psi$ for $|pa(X_i)| \leq k$:
    $p(pa(X_i)|\prec_l)$ for all $X_i$
    $p(Y \notin pa(X_i)|\prec_l)$ for $Y \prec_l X_i$;
    Compute $p(\prec_l |D_N)$;
    Construct order conditional MBG-Subspace$(\Pi, \Psi, R, \rho)$=$\Phi$
    $S^S$=Search($\Phi$,$L^S$, $\rho^S$);
    **for all** $mbg \in S^S$ **do**
      **if** $mbg \notin \mathcal{T}$ **then**
        Insert$(\mathcal{T}, mbg)$
    **if** $L^T < |\mathcal{T}|$ **then**
      $\mathcal{T}$=PruneToHPD($\mathcal{T}$,$L^T$);
    **for all** $mbg \in \mathcal{T}$ **do**
      $\hat{p}(mbg|D_N) += p(mbg|\prec_l, D_N)$;
  Report K MAP MBGs from $\mathcal{T}$;
  Report K' MAP MBs using the MBGs in $\mathcal{T}$;

Parameters $R, \rho$ allow the restriction of the MBG subspace separately for each dimension to a less than $R$ values by requiring that the corresponding posteriors are above the $\exp(-\rho)$ ratio of the respective MAP value. The uniform-cost search starts from the order conditional MAP MBG, and stops after the expansion of $L^S$ number of states or if the most probable MBG in its search list drops below $\exp(\rho^S)$ ratio of the order conditional posterior of the starting MBG.

Generally, the expansion phase has high computational costs, but for large $L^T$ the update of the MBGs in $\mathcal{T}$ is high as well. In order to maintain tractability the usage of more refined

methods such as partial updating are required. Within the explored OC domain however the full, exact update has acceptable costs if the size of the estimated MBG set is $L^T \in [10^5, 10^6]$. This $L^T$ ensures that the newly inserted MBGs are not pruned before their estimates can reliable indicate their high-scoring potential and still allows an exact update. In larger domains this balance can be different and the analysis of this question in general is for future research.

The analysis of the MBG space showed that the conditional posteriors of the ranked parental sets after rank 10 are negligible, so subsequently we will report results using values $R = 20, \rho = 4$ and $L^S = 10^4, \rho^S = 10^{-6}$. Note that the expansion with the $L^S$ conditionally most probable MBGs in each step does not guarantee that the $L^S$ most probable MBGs are estimated, not even the MAP MBG.

## 6 Results

We used a data set consisting of 35 discrete variables and 782 complete cases related to the preoperative diagnosis of ovarian cancer (see (Antal et al., 2004)).

First we report the estimation-selection of MB features for the central variable *Pathology*. We applied the heuristic deterministic search-estimation method in the inner cycle of the MCMC method. The length of the burn-in and MCMC simulation was 10000, the probability of the pairwise replace operator was 0.8, the parameter prior was the $BD_{eu}$ and the structure prior was uniform prior for the parental set sizes (Friedman and Koller, 2000). The maximum number of parents was 4 (the posteriors of larger sets are insignificant). For preselected high-scoring MB values after 10000 burn-in the single-chain convergence test from Geweke comparing averages has z-score approximately 0.5, the R value of the multiple-chain method of Gelman-Rubin with 5 chains drops below 1.05 (Gamerman, 1997; Gelman et al., 1995). The variances of the MCMC estimates of these preselected test feature values drop below $10^{-2}$. We also applied the deterministic search-estimation method for a single ordering, because a total ordering of the variables was available from an expert. Fig. 1 reports the estimated posteriors of the MAP MB sets for *Pathology* with their MBM-based approximated values assuming the independence of the MBM values and Table 1 shows the members of the MB sets. Note that the two monotone decreasing curves correspond to independent rankings, one for the expert's total ordering and one for the unconstrained case. It also reports the MB set spanned by a prior BN specified by the expert ($E$), the MB set spanned by the MAP BN ($BN$) and the set spanned by the MAP MBG ($MBG$) (see Eq. 6). Furthermore we generated another reference set ($LR$) from a conditional standpoint using the logistic regression model class and the SPSS 14.0 software with the default setting for forward model construction (Hosmer and Lemeshow, 2000). $MB_p$ reports the result of the deterministic select-estimate method using the total ordering of the expert and the $MB_1, MB_2, MB_3$ report the result of the unconstrained ordering-based MCMC with deterministic select-estimate method. Variables $FamHist$,$CycleDay$, $HormTherapy$, $Hysterectomy$, $Parity$ PMenoAge are never selected and the variables $Volume$, $Ascites$, $Papillation$, $PapFlow$, $CA125$, $WallRegularity$ are always selected, so they are not reported.

The MBM-based approximation performs relatively well, particularly w.r.t. ranking in the case of the expert's ordering $\prec_0$, but it performs poorly in the unconstrained case both w.r.t. estimations and ranks (see the difference of $M_p$ set to $M_1$ w.r.t. variables $Age$, $Meno$, $PI$, $TAMX$, $Solid$).

We compared the $MBG(Y, G^{MAP})$ and $MB(Y, G^{MAP})$ feature values defined by the MAP BN structure $G^{MAP}$ against the MAP MBG feature value $MBG(Y)^{MAP}$ and the MAP MB feature value $MB(Y)^{MAP}$ including the MB feature value defined by the MAP MBG feature value $MB(Y, MBG(Y)^{MAP})$

Table 1: Markov blanket sets of Pathology among the thirty-five variables.

| | E | LR | $BN$ | $MG$ | $MB_p$ | $MB_1$ | $MB_2$ | $MB_3$ |
|---|---|---|---|---|---|---|---|---|
| Age | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| Meno | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| PMenoY | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| PillUse | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bilateral | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| Pain | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Fluid | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Septum | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| ISeptum | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| PSmooth | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Loc. | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| Shadows | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| Echog. | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| ColScore | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| PI | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| RI | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| PSV | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| TAMX | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| Solid | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| FHBrCa | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| FHOvCa | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

$$
\begin{aligned}
G^{MAP} &= \arg\max_{G} p(G|D_N) \qquad (6)\\
MBG(Y)^{MAP} &= \arg\max_{mbg(Y)} p(mbg(Y)|D_N)\\
MB(Y)^{MAP} &= \arg\max_{mb(Y)} p(mb(Y)|D_N)
\end{aligned}
$$

We performed the comparison using the best BN structure found in the MCMC simulation. The MAP MBG feature value $MBG(Y)^{MAP}$ differed significantly from the MAP domain model, because of an additional *Age* and *Fluid* variables in the domain model. The MAP MB feature value $MB(Y)^{MAP}$ similarly differs from the MB sets defined by the MAP domain models for example w.r.t. the vascularization variables such as $PI$. Interestingly, the MAP MB feature value also differs from the MB feature value defined by the MAP MBG feature value $MB(Y, MBG(Y)^{MAP})$, for example w.r.t. $TAMX$, *Solid* variables. In conclusion these results together with the comparison against the simple feature-based analysis such as MBM-based analysis reported in Fig. 1, show the relevance of the complex feature-based analysis.

We also constructed an offline probabilistic knowledge base containing $10^4$ MAP MBGs. It

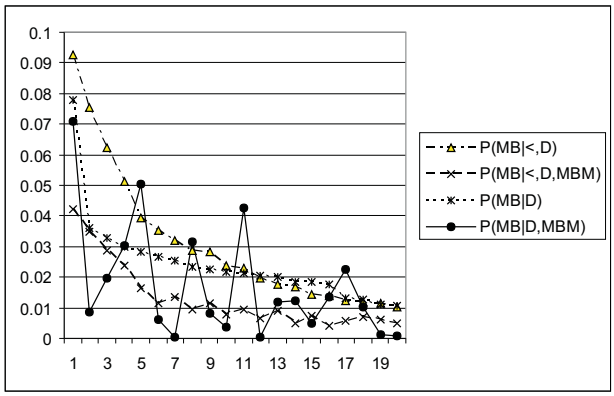



Figure 1: The ranked posteriors and their MBM-based approximations of the 20 most probable $MB(Pathology)$ sets for the single/unconstrained orderings.

is connected with the annotated BN knowledge base defined in Def. 1, which allows an offline exploration of the domain from the point of view of conditional modeling. The histogram of the number of parameters and inputs for the MBGs using only the fourteen most relevant variables are reported in Fig. 2.

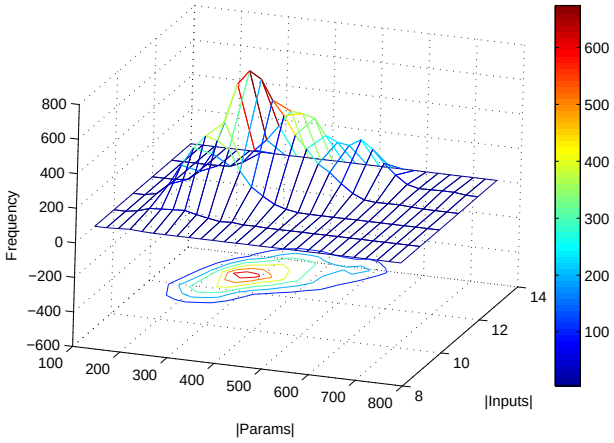



Figure 2: The histogram of the number of parameters and inputs of the MAP MBGs.

## 7 Conclusion

In the paper we presented a Bayesian approach for complex BN features, for the so-called Markov Blanket set and the Markov Blanket

Graph features. We developed and applied a select-estimate algorithm using ordering-based MCMC, which uses the efficiently computable order conditional posterior of the MBG feature and the proposed MBG space. The comparison of the most probable MB and MBG feature values with simple feature based approximations and with complete domain modeling showed the separate significance of the analysis based on these complex BN features in the investigated medical domain. The proposed algorithm and the offline knowledge base in the introduced probabilistic annotated BN knowledge base context allows new types of analysis and fusion of expertise, data and literature.

**Acknowledgements**

We thank T. Dobrowiecki for his helpful comments. Our research was supported by grants from Research Council KUL: IDO (IOTA Oncology, Genetic networks).

## References

P. Antal, G. Fannes, Y. Moreau, D. Timmerman, and B. De Moor. 2004. Using literature and data to learn Bayesian networks as clinical models of ovarian tumors. *AI in Med.*, 30:257–281. Special issue on Bayesian Models in Med.

W. L. Buntine. 1991. Theory refinement of Bayesian networks. In *Proc. of the 7th Conf. on Uncertainty in Artificial Intelligence* , pages 52–60. Morgan Kaufmann.

G. F. Cooper and E. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

N. Friedman and D. Koller. 2000. Being Bayesian about network structure. In *Proc. of the 16th Conf. on Uncertainty in Artificial Intelligence*, pages 201–211. Morgan Kaufmann.

N. Friedman, M. Goldszmidt, and A. Wyner. 1999. Data analysis with bayesian networks: A Bootstrap approach. In *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence*, pages 196–205. Morgan Kaufmann.

D. Gamerman. 1997. *Markov Chain Monte Carlo*. Chapman & Hall, London.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. 1995. *Bayesian Data Analysis*. Chapman & Hall, London.

C. Glymour and G. F. Cooper. 1999. *Computation, Causation, and Discovery*. AAAI Press.

D. Heckerman, D. Geiger, and D. Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

D. Heckermann, C. Meek, and G. Cooper. 1997. A bayesian aproach to causal discovery. *Technical Report*, MSR-TR-97-05.

D. W. Hosmer and S. Lemeshow. 2000. *Applied Logistic Regression*. Wiley & Sons, Chichester.

R. Kohavi and G. H. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324.

D. Koller and M. Sahami. 1996. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292.

D. Madigan, S. A. Andersson, M. Perlman, and C. T. Volinsky. 1996. Bayesian model averaging and model selection for markov equivalence classes of acyclic digraphs. *Comm.Statist. Theory Methods*, 25:2493–2520.

S. Mani and G. F. Cooper. 2001. A simulation study of three related causal data mining algorithms. In *International Workshop on Artificial Intelligence and Statistics*, pages 73–80. Morgan Kaufmann, San Francisco, CA.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, CA.

J. Pearl. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

D. Pe'er, A. Regev, G. Elidan, and N. Friedman. 2001. Inferring subnetworks from perturbed expression profiles. *Bioinformatics, Proceedings of ISMB 2001*, 17(Suppl. 1):215–224.

C. Silverstein, S. Brin, R. Motwani, and J. D. Ullman. 2000. Scalable techniques for mining causal structures. *Data Mining and Knowledge Discovery*, 4(2/3):163–192.

D. J. Spiegelhalter and S. L. Lauritzen. 1990. Sequential updating of conditional probabilities on directed acyclic graphical structures. *Networks*, 20(.):579–605.

P. Spirtes, C. Glymour, and R. Scheines. 2001. *Causation, Prediction, and Search*. MIT Press.

I. Tsamardinos and C. Aliferis. 2003. Towards principled feature selection: Relevancy,filters, and wrappers. In *Proc. of the Artificial Intelligence and Statistics*, pages 334–342.

# Literature Mining using Bayesian Networks

Péter Antal and András Millinghoffer
Department of Measurement and Information Systems
Budapest University of Technology and Economics

**Abstract**

In biomedical domains, free text electronic literature is an important resource for knowledge discovery and acquisition, particularly to provide a priori components for evaluating or learning domain models. Aiming at the automated extraction of this prior knowledge we discuss the types of uncertainties in a domain with respect to causal mechanisms, formulate assumptions about their report in scientific papers and derive generative probabilistic models for the occurrences of biomedical concepts in papers. These results allow the discovery and extraction of latent causal dependency relations from the domain literature using minimal linguistic support. Contrary to the currently prevailing methods, which assume that relations are sufficiently formulated for linguistic methods, our approach assumes only the report of causally associated entities without their tentative status or relations, and can discover new relations and prune redundancies by providing a domain-wide model. Therefore the proposed Bayesian network based text mining is an important complement to the linguistic approaches.

## 1 Introduction

Rapid accumulation of biological data and the corresponding knowledge posed a new challenge of making this voluminous, uncertain and frequently inconsistent knowledge accessible. Despite recent trends to broaden the scope of formal knowledge bases in biomedical domains, free text electronic literature is still the central repository of the domain knowledge. This central role will probably be retained in the near future, because of the rapidly expanding frontiers. The extraction of explicitly stated or the discovery of implicitly present latent knowledge requires various techniques ranging from purely linguistic approaches to machine learning methods. In the paper we investigate a domain-model based approach to statistical inference about dependence and causal relations given the literature using minimal linguistic preprocessing. We use Bayesian Networks (BNs) as causal domain models to introduce generative models of publication, i.e. we examine the relation of domain models and generative models of the corresponding literature.

In a wider sense our work provides support to statistical inference about the structure of the domain model. This is a two-step process, which consists of the reconstruction of the beliefs in mechanisms from the literature by model learning and their usage in a subsequent learning phase. Here, the Bayesian framework is an obvious choice. Earlier applications of text mining provided results for the domain experts or data analysts, whereas our aim is to go one step further and use the results directly in the statistical learning of the domain models.

The paper is organized as follows. Section 2 presents a unified view of the literature, the data and their models. In Section 3 we review the types of uncertainties in biomedical domains from a causal, mechanism-oriented point of view. In Section 4 we summarize recent approaches to information extraction and literature mining based on natural language processing (NLP) and "local" analysis of occurrence patterns. In Section 5 we propose generative probabilistic models for the occurrences of biomedical concepts in scientific papers. Section 6 presents textual aspects of the application

domain, the diagnosis of ovarian cancer. Section 7 reports results on learning BNs given the literature.

## 2 Fusion of literature and data

The relation of experimental data $D_N$, probabilistic causal domain models formalized as BNs $(G, \theta)$, domain literature $D^L_{N'}$ and models of publication $(G^L, \theta^L)$ can be approached at different levels. For the moment, let us assume that probabilistic models are available describing the generation of observations $P(D_N|(G, \theta))$ and literature $P(D^L_{N'}|(G^L, \theta^L))$. This latter may include stochastic grammars for modeling the linguistic aspects of the publication, however, we will assume that the literature has a simplified agrammatical representation and the corresponding generative model can be formalized as a BN $(G^L, \theta^L)$ as well.

The main question is the relation of $P(G, \theta)$ and $P(G^L, \theta^L)$. In the most general approach the hypothetical posteriors $P(G, \theta|D_N, \xi_i)$ expressing personal beliefs over the domain models conditional on the experiments and the personal background knowledge $\xi_i$ determine or at least influence the parameters of the model $(G^L, \theta^L)$ in $P(D^L_{N'}|(G^L, \theta^L), \xi_i)$.

The construction or the learning of a full-fledged decision theoretic model of publication is currently not feasible regarding the state of quantitative modeling of scientific research and publication policies, not to mention the cognitive and even stylistic aspects of explanation, understanding and learning (Rosenberg, 2000). In a severely restricted approach we will focus only on the effect of the belief in domain models $P(G, \theta)$ on that in publication models $P(G^L, \theta^L)$. We will assume that this transformation is "local", i.e. there is a simple probabilistic link between the model spaces, specifically between the structure of the domain model and the structure and parameters of the publication model $p(G^L, \theta^L|G)$. Probabilistically linked model spaces allow the computation of the posterior over domain models given the literature(!) data as:

$$P(G|D^L_{N'}) = \frac{P(G)}{P(D^L_{N'})} \sum_{G^L} P(D^L_{N'}|G^L)P(G^L|G).$$

The formalization $(D_N \leftarrow G \rightarrow G^L \rightarrow D^L_{N'})$ also allows the computation of the posterior over the domain models given both clinical and the literature(!) data as:

$$\begin{aligned} P(G|D_N, D^L_{N'}) &= P(G)\frac{P(D^L_{N'}|G)}{P(D^L_{N'})}\frac{P(D_N|G)}{P(D_N|D^L_{N'})} \\ &\propto P(G)P(D_N|G)\sum_{G^L} P(D^L_{N'}|G^L)P(G^L|G), \end{aligned}$$

The order of the factors shows that the prior is first updated by the literature data, then by the clinical data. A considerable advantage of this approach is the integration of literature and clinical data at the lowest level and not through feature posteriors, i.e. by using literature posteriors in feature-based priors for the (clinical) data analysis (Antal et al., 2004).

We will assume that a bijective relation exists between the domain model structures $G$ and the publication model structures $G^L$ $(\mathcal{T}(G) = G^L)$, whereas the parameters $\theta^L$ may encode additional aspects of publication policies and explanation. We will focus on the logical link between the structures, where the posterior given the literature and possibly the clinical data is:

$$\begin{aligned} &P(G|D_N, D^L_{N'}, \xi) \qquad (1) \\ &\propto \quad P(G|\xi)P(D_N|G)P(D^L_{N'}|\mathcal{T}(G)). \end{aligned}$$

This shows the equal status of the literature and the clinical data. In integrated learning from heterogeneous sources however, the scaling of the sources is advisable to express our confidence in them.

## 3 Concepts, associations, causation

Frequently a biomedical domain can be characterized by a dominant type of uncertainty w.r.t the causal mechanisms. Such types of uncertainty show certain sequentiality described below, related to the development of biomedical knowledge, though a strictly sequential view is clearly an oversimplification.

(1) *Conceptual phase*: Uncertainty over the domain ontology, i.e. the relevant entities.

(2) *Associative phase*: Uncertainty over the association of entities, reported in the literature as indirect, associative hypotheses, frequently as clusters of entities. Though we accept the general view of causal relations behind associations, we assume that the exact causal functions and direct relations are unknown.

(3) *Causal relevance phase*: (Existential) uncertainty over causal relations (i.e. over mechanisms). Typically, direct causal relations are theoretized as processes and mechanisms.

(4) *Causal effect phase*: Uncertainty over the strength of the autonomous mechanisms embodying the causal relations.

In this paper we assume that the target domain is already in the Associative or Causal phase, i.e. that the entities are more or less agreed, but their causal relations are mostly in the discovery phase. This holds in many biomedical domains, particularly in those linking biological and clinical levels. There the Associative phase is a crucial but lengthy knowledge accumulation process, where wide range of research methods is used to report associated pairs or clusters of the domain entities. These methods admittedly produce causally oriented associative relations which are partial, biased and noisy.

## 4 Literature mining

Literature mining methods can be classified into bottom-up (pairwise) and top-down (domain model based) methods. Bottom-up methods attempt to identify individual relationships and the integration is left to the domain expert. Linguistic approaches assume that the individual relations are sufficiently known, formulated and reported for automated detection methods. On the contrary, top-down methods concentrate on identifying consistent domain models by analyzing jointly the domain literature. They assume that mainly causally associated entities are reported with or without tentative relations and direct structural knowledge. Their linguistic formulation is highly variable, not conforming to simple grammatical characterization. Consequently top-down methods typically use agrammatical text representations and minimal linguistic support. They autonomously prune the redundant, inconsistent, indirect relations by evaluating consistent domain models and can deliver results in domains already in the *Associative* phase.

Until recently mainly bottom-up methods have been analyzed in the literature: *linguistic approaches* extract explicitly stated relations, possibly with qualitative ratings (Proux et al., 2000; Hirschman et al., 2002); *co-occurrence analysis* quantifies the pairwise relations of variables by their relative frequency (Stapley and Benoit, 2000; Jenssen et al., 2001); *kernel similarity analysis* uses the textual descriptions or the occurrence patterns of variables in publications to quantify their relation (Shatkay et al., 2002); Swanson and Smalheiser (1997) discover relationships through the heuristic pattern analysis of citations and co-occurrences; in (Cooper, 1997) and (Mani and Cooper, 2000) local constraints were applied to cope with possible hidden confounders, to support the discovery of causal relations; *joint statistical analysis* in (Krauthammer et al., 2002) fits a generative model to the temporal pattern of corroborations, refutations and citations of individual relations to identify "true" statements. The top-down method of the *joint statistical analysis* of de Campos (1998) learns a restricted BN thesaurus from the occurrence patterns of words in the literature. Our approach is closest to this and those of Krauthammer et al. and Mani.

The reconstruction of informative and faithful priors over domain mechanisms or models from research papers is further complicated by the *multiple aspects of uncertainty* about the existence, scope (conditions of validity), strength, causality (direction), robustness for perturbation and relevance of mechanism and the *incompleteness of reported relations*, because they are assumed to be well-known parts of *common sense* knowledge or of the *paradigmatic* already reported knowledge of the community.

## 5 BN models of publications

Considering (biomedical) abstracts, we adopt the central role of causal understanding and explanation in scientific research and publication (Thagard, 1998). Furthermore, we assume that the contemporary (collective) uncertainty over mechanisms is an important factor influencing the publications. According to this causal stance, we accept the 'causal relevance' interpretation, more specifically the 'explained' (explanandum) and 'explanatory' (explanans), in addition, we allow the 'described' status. This is appealing, because in the assumed causal publications both the name occurrence and the preprocessing kernel similarity method (see Section 6) express the presence or relevance of the concept corresponding to the respective variable. This implicitly means that we assume that publications contain either descriptions of the domain concepts without considering their relations or the occurrences of entities participating in known or latent causal relations. We assume that there is only one causal mechanism for each parental set, so we will equate a given parental set and the mechanism based on it.

Furthermore, we assume that mainly positive statements are reported and we treat negation and refutation as noise, and that exclusive hypotheses are reported, i.e. we treat alternatives as one aggregated hypothesis. Additionally, we presume that the dominant type of publications are causally ("forward") oriented. We attempt to model the transitive nature of causal explanation over mechanisms, e.g. that causal mechanisms with a common cause or with a common effect are surveyed in an article, or that subsequent causal mechanisms are tracked to demonstrate a causal chain. On the other hand, we also have to model the lack of transitivity, i.e. the incompleteness of causal explanations, e.g. that certain variables are assumed as explanatory, others as potentially explained, except for survey articles that describe an overall domain model. Finally, we assume that the reports of the causal mechanisms and the univariate descriptions are independent of each other.

### 5.1 The intransitive publication model

The first generative model is a two-layer BN. The upper-layer variables represent the pragmatic functions (described or explanandum) of the corresponding concepts, while lower-layer variables represent their observable occurrences (described, explanatory or explained). Upper-layer variables can be interpreted as the intentions of the authors or as the property of the given experimental technique. We assume that lower-layer variables are influenced only by the upper-layer ones denoting the corresponding mechanisms, and not by any other external quantities, e.g. by the number of the reported entities in the paper. A further assumption is that the belief in a compound mechanism is the product of the beliefs in the pairwise dependencies. Consequently we use noisy-OR canonic distributions for the children in the lower layer. In a noisy-OR local dependency (Pearl, 1988), the edges can be labeled with a parameter, inhibiting the OR function, which can be interpreted also structurally as the probability of an implicative edge.

This model extends the atomistic, individual-mechanism oriented information extraction methods by supporting the joint learning of all the mechanisms, i.e. by the search for a domain-wide coherent model. However it still cannot model the dependencies between the reported associations, and the presence of hidden variables considerably increase the computational complexity of parameter and structure learning.

### 5.2 The transitive publication model

To devise a more advanced model, we relax the assumption of the independence between the variables in the upper layer representing the pragmatic functions, and we adapt the models to the bag-of-word representation of publications (see Section 6). Consequently we analyze the possible pragmatic functions corresponding to the domain variables, which could be represented by hidden variables. We assume here that the explanatory roles of a variable are not differentiated, and that if a variable is explained, then it can be explanatory for any other

variable. We assume also full observability of causal relevance, i.e. that the lack of occurrence of an entity in a paper means causal irrelevance w.r.t. the mechanisms and variables in the paper and not a neutral omission. These assumptions allow the merging of the explanatory, explained and described status with the observable reported status, i.e. we can represent them jointly with a single binary variable. Note that these assumptions remain tenable in case of report of experiments, where the pattern of relevancies has a transitive-causal bias.

These would imply that we can model only full survey papers, but the general, unconstrained multinomial dependency model used in the transitive BNs provides enough freedom to avoid this. A possible semantics of the parameters of a binary, transitive literature BN can be derived from a causal stance that the presence of an entity $X_i$ is influenced only by the presence of its potential explanatory entities, i.e. its parents. Consequently, $P(X_i = 1|Pa_{X_i} = pa_{x_i})$ can be interpreted as the belief that the present parental variables can explain the entities $X_i$ ($Pa_{X_i}$ denotes the parents of $X_i$ and $Pa_{X_i} \rightarrow X_i$ denotes the parental substructure). In that way the parameters of a complete network can represent the priors for parental sets compatible with the implied ordering:

$$P(X_i = 1|Pa_{X_i} = pa_{X_i}) = P(Pa_{X_i} = pa_{X_i}) \tag{2}$$

where for notational simplicity $pa(X_i)$ denotes both the parental set and a corresponding binary representation.

The multinomial model allows entity specific modifications at each node, combined into the parameters of the conditional probability model that are independent of other variables (i.e. unstructured noise). This permits the modeling of the description of the entities ($P(X_i^{\mathrm{D}})$), the beginning of the transitive scheme of causal explanation ($P(X_i^{\mathrm{B}})$) and the reverse effect of interrupting the transitive scheme ($P(X_i^{\mathrm{I}})$). These auxiliary variables model simplistic interventions, i.e. authors' intentions about publishing an observational model. Note that a "backward" model corresponding to an effect-to-cause or diagnostic interpretation and explanation method has a different structure with opposite edge directions.

In the Bayesian framework, there is a structural uncertainty also, i.e. uncertainty over the structure of the generative models (literature BNs) themselves. So to compute the probability of a parental set $Pa_{X_i} = pa_{X_i}$ given a literature data set $D^L_{N'}$, we have to average over the structures using the posterior given the literature data:

$$\begin{aligned} P(Pa_{X_i} = pa_{X_i}|D^L_{N'}) & \qquad (3) \\ = \sum_{(pa_{X_i} \rightarrow X_i) \subset G} & P(X_i = 1|pa_{X_i}, G) P(G|D^L_{N'}) \\ \approx \sum_{G} & 1((pa_{X_i} \rightarrow X_i) \subset G) P(G|D^L_{N'}) \qquad (4) \end{aligned}$$

Consequently, the result of learning BNs from the literature can be multiple, e.g. using a maximum a posteriori (MAP) structure and the corresponding parameters, or the posterior over the structures (Eq. 3). In the first case, the parameters can be interpreted structurally and converted into a prior for a subsequent learning. In the latter case, we neglect the parametric information focusing on the structural constraints, and transform the posterior over the literature network structures into a prior over the structures of the real-world BNs (see Eq. 1).

## 6 The literature data sets

For our research we used the same collection of abstracts as that described in (Antal et al., 2004), which was a preliminary work using pairwise methods. The collection contains 2256 abstracts about ovarian cancer, mostly between 1980 and 2002. Also a name, a list of synonyms and a text kernel is available for each domain variable. The presence of the name (and synonyms) of a variable in documents is denoted with a binary value. Another binary representation of the publications is based on the kernel documents:

$$R^K_{ij} = \begin{cases} 1 \text{ if } 0.1 < \mathrm{sim}(k_j, d_i) \\ 0 \text{ else} \end{cases}, \tag{5}$$

which expresses the relevance of kernel document $k_j$ to document $d_i$ using the 'term

frequency-inverse document frequency' (TF-IDF) vector representation and the cosine similarity metric (Baeza-Yates and Ribeiro-Neto, 1999). We use the term *literature data* to denote both binary representations of the relevance of concepts in publications, usually denoted with $D^L_{N'}$ (containing $N'$ publications).

## 7 Results

The structure learning of the transitive model is achieved by an exhaustive evaluation of parental sets up to 4 variables followed by the K2 greedy heuristics using the $BD_{eu}$ score (Heckerman et al., 1995) and an ordering of the variables from an expert, in order to be compatible with the learning of the intransitive model. The structure learning of the two-layer model has a higher computational cost, because the evaluation of a structure requires the optimization of parameters, which can be performed e.g. by gradient-descent algorithms. Because of the use of the "forward" explanation scheme, only those variables in the upper layer can be the parents of an external variable that succeed it in the causal order. Note that beside the optional parental edges for the external variables, we always force a deterministic edge from the corresponding non-external variable. During the parameter learning of a fixed network structure the non-zero inhibitory parameters of the lower layer variables are adjusted according to a gradient descent method to maximize the likelihood of the data (see (Russell et al., 1995)). After having found the best structure, according to its semantics, it is converted into a flat, real-world structure without hidden variables. This conversion involves the merging of the corresponding pairs of nodes of the two layers, and then reverting the edges (since in the explanatory interpretation effects precede causes).

We compared the trained models to the expert model using a quantitative score based on the comparison of the pairwise relations in the model, which are defined w.r.t. the causal interpretation as follows (Cooper and Yoo, 1999; Wu et al., 2001): *Causal edge (E)* An edge between the nodes. *Causal path (P)* A directed path linking nodes. *(Pure) Confounded (C)* The two nodes have a common ancestor. The relation is pure, if there is no edge or path between the nodes. *Independent (I)* None of the previous (i.e. there is no causal connection).

The difference between two model structures can be represented in a matrix containing the number of relations of a given type in the expert model and in the trained model (the type of the relation in the expert model is the row index and the type in the trained model is the column index). These matrices (i.e. the comparison of the transitive and the intransitive models to the expert's) are shown in Table 1. Scalar scores can be derived from this matrix, to evaluate the goodness of the trained model, the standard choice is to sum the elements with different weights (Cooper and Yoo, 1999; Wu et al., 2001). One possibility e.g. if we take the sum of the diagonal elements as a measure of similarity. By this comparison, the intransitive model achieves 148 points, while the transitive 358, so the transitive reconstructs more faithfully the underlying structure. Particularly important is the $(E, E)$ element according to which 52 of the 120 edges of the expert model remains in the transitive model, on the contrary the intransitive model preserves only 36 edges. Similarly the independent relations of the expert model are well respected by both models.

Table 1: Causal comparison of the intransitive and the transitive domain models (columns with 'i' and 't' in the subscript, respectively) to the expert model (rows).

| | $I_i$ | $C_i$ | $P_i$ | $E_i$ | $I_t$ | $C_t$ | $P_t$ | $E_t$ |
|---|---|---|---|---|---|---|---|---|
| $I$ | 12 | 0 | 0 | 0 | 0 | 4 | 2 | 6 |
| $C$ | 106 | 20 | 2 | 4 | 4 | 90 | 26 | 12 |
| $P$ | 756 | 72 | 80 | 18 | 188 | 460 | 216 | 62 |
| $E$ | 70 | 6 | 8 | 36 | 6 | 38 | 24 | 52 |

Another score, which penalizes only the incorrect identification of independence (i.e. those and only those weights have a value of 1 which belong to the elements $(I, .)$ or $(., I)$, the others are 0), gives a score 210 and 932 for the transitive model and the intransitive respectively.

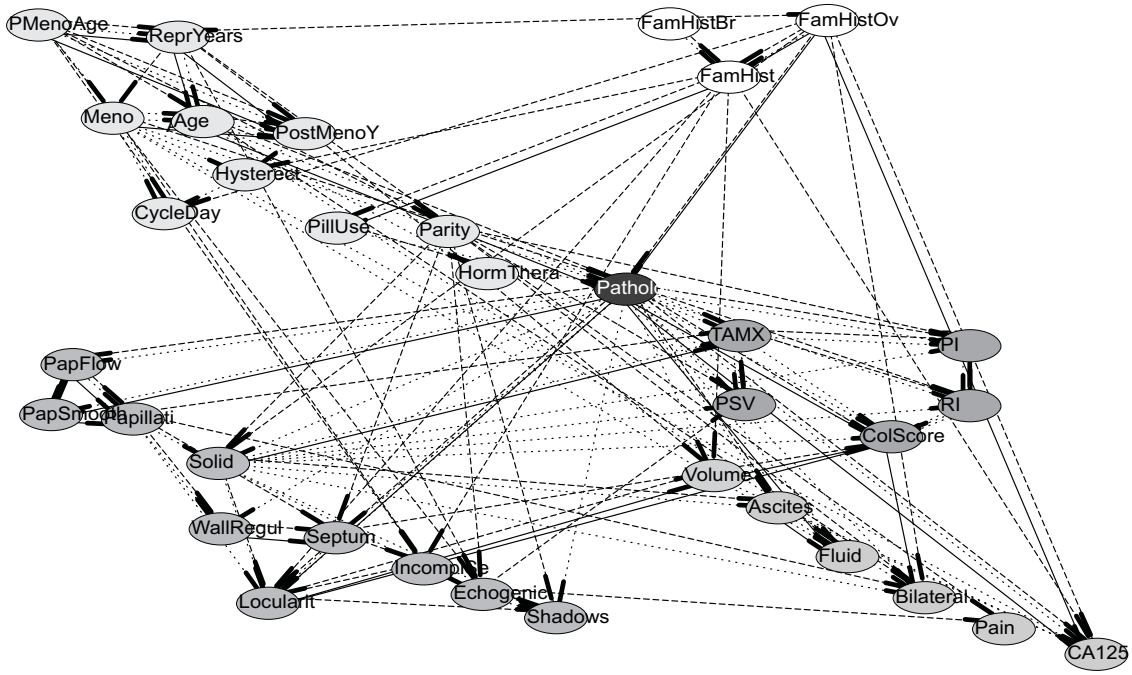



Figure 1: The expert-provided (dotted), the MAP transitive (dashed) and the intransitive (solid) BNs compatible with the expert's total ordering of the thirty-five variables using the literature data set ($PM_{R_3}^{COREL}$), the K2 noninformative parameter priors, and noninformative structure priors.

This demonstrates that the intransitive model is extremely conservative in comparison with both the other learning method and with the knowledge of the expert, it is only capable of detecting the most important edges; note that the proportion of its false positive predictions regarding the edges is only 38% while in the transitive model it is 61%.

Furthermore, we investigated the Bayesian learning of BN features, particularly using the temporal sequence of the literature data sets. An important feature indicating relevance between two variables is the so-called 'Markov Blanket Membership' (Friedman and Koller, 2000). We have examined the temporal characteristics of the posterior of this relation between a target variable 'Pathology' and the other ones using the approximation in Eq. 4. This feature is a good representative for the diagnostic importance of variables according to the community. We have found four types of variables: the posterior of the relevance increasing in time fast or slowly, decreasing slowly or fluctuating. Figure 2 shows examples for variables with a slow rising in time.

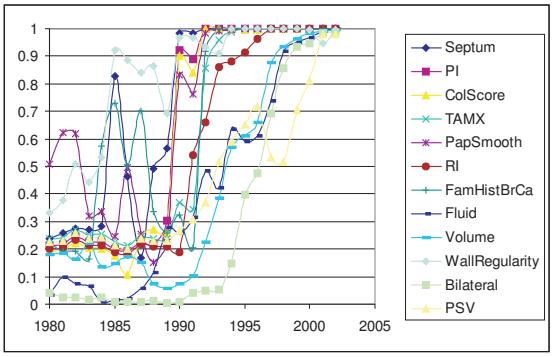



Figure 2: The probability of the relation Markov Blanket Membership between Pathology and the variables with a slow rise.

## 8 Conclusion

In the paper we proposed generative BN models of scientific publication to support the construction of real-world models from free-text literature. The advantage of this approach is its

domain model based foundation, hence it is capable of constructing coherent models by autonomously pruning redundant or inconsistent relations. The preliminary results support this expectation. In the future we plan to use the evaluation methodology applied there including rank based performance metrics and to investigate the issue of negation and refutation particularly through time.

## References


P. Antal, G. Fannes, Y. Moreau, D. Timmerman, and B. De Moor. 2004. Using literature and data to learn Bayesian networks as clinical models of ovarian tumors. *Artificial Intelligence in Medicine*, 30:257–281. Special issue on Bayesian Models in Medicine.

R. Baeza-Yates and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. ACM Press, New York.

G. F. Cooper and C. Yoo. 1999. Causal discovery from a mixture of experimental and observational data. In *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence (UAI-1999)*, pages 116–125. Morgan Kaufmann.

G. Cooper. 1997. A simple constraint-based algorithm for efficiently mining observational databases for causal relationships. *Data Mining and Knowledge Discovery*, 2:203–224.

L. M. de Campos, J. M. Fernández, and J. F. Huete. 1998. Query expansion in information retrieval systems using a Bayesian network-based thesaurus. In Gregory Cooper and Serafin Moral, editors, *Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence (UAI-1998)*, pages 53–60. Morgan Kaufmann.

N. Friedman and D. Koller. 2000. Being Bayesian about network structure. In Craig Boutilier and Moises Goldszmidt, editors, *Proc. of the 16th Conf. on Uncertainty in Artificial Intelligence(UAI-2000)*, pages 201–211. Morgan Kaufmann.

D. Geiger and D. Heckerman. 1996. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82:45–74.

D. Heckerman, D. Geiger, and D. Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

L. Hirschman, J. C. Park, J. Tsujii, L. Wong, and C. H. Wu. 2002. Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, 18:1553–1561.

T. K. Jenssen, A. Laegreid, J. Komorowski, and E. Hovig. 2001. A literature network of human genes for high-throughput analysis of gene expression. *Nature Genetics*, 28:21–28.

M. Krauthammer, P. Kra, I. Iossifov, S. M. Gomez, G. Hripcsak, V. Hatzivassiloglou, C. Friedman, and A. Rzhetstky. 2002. Of truth and pathways: chasing bits of information through myriads of articles. *Bioinformatics*, 18:249257.

S. Mani and G. F. Cooper. 2000. Causal discovery from medical textual data. In *AMIA Annual Symposium*.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, CA.

D. Proux, F. Rechenmann, and L. Julliard. 2000. A pragmatic information extraction strategy for gathering data on genetic interactions. In *Proc. of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'2000), La-Jolla, California*, pages 279–285.

A. Rosenberg. 2000. *Philosophy of Science: A contemporary introduction*. Routledge.

S. J. Russell, J. Binder, D. Koller, and K. Kanazawa. 1995. Local learning in probabilistic networks with hidden variables. In *IJCAI*, pages 1146–1152.

H. Shatkay, S. Edwards, and M. Boguski. 2002. Information retrieval meets gene analysis. *IEEE Intelligent Systems*, 17(2):45–53.

B. Stapley and G. Benoit. 2000. Biobibliometrics: Information retrieval and visualization from co-occurrences of gene names in medline asbtracts. In *Proc. of Pacific Symposium on Biocomputing (PSB00)*, volume 5, pages 529–540.

D. R. Swanson and N. R. Smalheiser. 1997. An interactive system for finding complementary literatures: a stimulus to scientific discovery. *Artificial Intelligence*, 91:183–203.

P. Thagard. 1998. Explaining disease: Correlations, causes, and mechanisms. *Minds and Machines*, 8:61–78.

X. Wu, P. Lucas, S. Kerr, and R. Dijkhuizen. 2001. Learning bayesian-network topologies in realistic medical domains. In *In Medical Data Analysis: Second International Symposium, ISMDA*, pages 302–308. Springer-Verlag, Berlin.

# Locally specified credal networks

Alessandro Antonucci and Marco Zaffalon
Istituto "Dalle Molle" Di Studi Sull'Intelligenza Artificiale
CH-6928 Manno (Lugano), Switzerland
{alessandro,zaffalon}@idsia.ch

**Abstract**

*Credal networks* are models that extend Bayesian nets to deal with imprecision in probability, and can actually be regarded as sets of Bayesian nets. Evidence suggests that credal nets are a powerful means to represent and deal with many important and challenging problems in uncertain reasoning. We give examples to show that some of these problems can only be modelled by credal nets called *non-separately specified*. These, however, are still missing a graphical representation language and solution algorithms. The situation is quite the opposite with separately specified credal nets, which have been the subject of much study and algorithmic development. This paper gives two major contributions. First, it delivers a new graphical language to formulate any type of credal network, both separately and non-separately specified. Second, it shows that *any* non-separately specified net represented with the new language can be easily transformed into an equivalent separately specified net, defined over a larger domain. This result opens up a number of new perspectives and concrete outcomes: first of all, it immediately enables the existing algorithms for separately specified credal nets to be applied to non-separately specified ones.

## 1 Introduction

We focus on credal networks (Section 3) (Cozman, 2005), which are a generalization of Bayesian nets. The generalization is achieved by relaxing the requirement that the conditional mass functions of the model be precise: with credal nets each of them is only required to belong to a closed convex set. Closed convex sets of mass functions are also known as *credal sets* after Levi (Levi, 1980). Using credal sets in the place of mass functions makes credal networks an *imprecise probability* model (Walley, 1991). It can be shown that a credal network is equivalent to a *set of Bayesian nets* with the same graph.

An important question is whether or not all credal networks can be represented in a way that emphasizes locality. The answer is positive if we restrict the attention to the most popular type of credal networks, those called *separately specified* (Section 4). In this case, each conditional mass function is allowed to vary in its credal set independently of the others. The representation is naturally local because there are no relationships between different credal sets. The question is more complicated with more general specifications of credal networks, which we call *non-separately specified*. The idea of non-separately specified credal nets is in fact to allow for relationships between conditional mass functions in different credal sets, which can also be far away in the net.

Although the idea of non-separately specified credal nets is relatively intuitive, it should be stressed that this kind of nets has been investigated very little: in fact, there has been no attempt so far to develop a general graphical language to describe them; and there is no algorithm to compute with them.[1] This appears

[1] An exception is the classification algorithm developed for the *naive credal classifier* (Zaffalon, 2001), but it is ad hoc for a very specific type of network. More generally speaking, it is not unlikely that some of the

to be an unfortunate gap in the literature as the non-separate specification seems to be the key to model many important problems, as illustrated in Section 6. Separately specified credal nets, on the other hand, have been the subject of much algorithmic development (Cozman, 2005).

In this paper we give two major contributions. First, we define a unified graphical language to locally specify credal networks in the general case (Section 5). The new representation is inspired, via the CCM transformation (Cano et al., 1994), by the formalism of influence diagrams, and more generally of decision graphs (Zhang et al., 1993). In this language the graph of a credal net is augmented with control nodes that express the relationships between different credal sets. We give examples to show that the new language provides one with a natural way to define non-separately specified nets; and we give a procedure to reformulate any separately specified net in the new language.

Second, we make a very simple observation (Section 7), which has surprisingly powerful implications: we show that for any credal network specified with the new language there is a separately specified credal network, defined over a larger domain, which is equivalent. The procedure to transform the former into the latter network is very simple, and takes only linear time. The key point is that this procedure can be used as a tool to "separate" the credal sets of non-separately specified nets. This makes it possible to model, by separately specified nets, problems formerly modelled by non-separately specified ones; and hence to use *any* (both exact and approximate) existing algorithm for separately specified nets to solve such problems.

Some comments on this result and perspectives for future developments are discussed in Section 8. The more technical parts of this paper are collected in Appendix A. ————-

existing algorithms for separately specified nets can be extended to special cases of non-separate specification, but we are not aware of any published work dealing with this issue.

## 2 Basic notation and Bayesian nets

Let us first define some notation and the fundamental notion of Bayesian network. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a collection of random variables, which take values in finite sets, and $\mathcal{G}$ a directed acyclic graph (DAG), whose nodes are associated to the variables in $\mathbf{X}$. For each $X_i \in \mathbf{X}$, $\Omega_{X_i}$ is the possibility space of $X_i$, $x_i$ a generic element of $\Omega_{X_i}$, $P(X_i)$ a mass function for $X_i$ and $P(x_i)$ the probability that $X_i = x_i$. The parents of $X_i$, according to $\mathcal{G}$, are denoted by the joint variable $\Pi_i$, whose possibility space is $\Omega_{\Pi_i}$. For each $\pi_i \in \Omega_{\Pi_i}$, $P(X_i|\pi_i)$ is the conditional mass function for $X_i$ given the joint value $\pi_i$ of the parents of $X_i$. This formalism is sufficient to properly introduce the following:

**Definition 1.** A Bayesian network (BN) over $\mathbf{X}$ is a pair $\langle \mathcal{G}, \mathbb{P} \rangle$ such that $\mathbb{P}$ is a set of conditional mass functions $P(X_i|\pi_i)$, one for each $X_i \in \mathbf{X}$ and $\pi_i \in \Omega_{\Pi_i}$.

We assume the *Markov condition* to make $\mathcal{G}$ represent probabilistic independence relations between the variables in $\mathbf{X}$: every variable is independent of its non-descendant non-parents conditional on its parents. Thus, a BN determines a joint mass function $P(\mathbf{X})$ according to the following factorization:

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\pi_i), \tag{1}$$

for each $\mathbf{x} \in \Omega_{\mathbf{X}}$, where for each $i = 1, \ldots, n$ the values $(x_i, \pi_i)$ are consistent with $\mathbf{x}$.

## 3 Credal sets and credal networks

Credal networks extend Bayesian nets to deal with imprecision in probability. This is obtained by means of closed convex sets of probability mass functions, which are called *credal sets* (Levi, 1980). We follow (Cozman, 2000) in considering only finitely generated credal sets, i.e., obtained as the convex hull of a finite number of mass functions. Geometrically, a credal set is a *polytope*. A credal set contains an infinite number of mass functions, but only a finite number of *extreme mass functions*: those corresponding to the *vertices* of the polytope, which

are, in general, a subset of the generating mass functions. A credal set over $X$ is denoted as $K(X)$. We similarly denote as $K(X|y)$ a *conditional credal set* over $X$ given a value $y$ of another random variable $Y$, i.e., a credal set of conditional mass functions $P(X|y)$. Given a *joint credal set* $K(X,Y)$, the *marginal credal set* for $X$ is the credal set $K(X)$ obtained by the point-wise marginalization of $Y$ from all the joint mass function $P(X,Y) \in K(X,Y)$. Finally, given a subset $\Omega'_X \subseteq \Omega_X$, a particularly important credal set for our purposes is the *vacuous credal set* for $\Omega'_X$, i.e., the set of all mass functions over $X$ assigning probability one to $\Omega'_X$. In the following we will use the well known fact that the vertices of such a credal set are the $|\Omega'_X|$ degenerate mass functions assigning probability one to the single elements of $\Omega'_X$.

**Definition 2.** A credal network (CN) over $\mathbf{X}$ is a pair $\langle \mathcal{G}, \{\mathbb{P}_1, \ldots, \mathbb{P}_m\}\rangle$ such that $\langle \mathcal{G}, \mathbb{P}_j\rangle$ is a Bayesian network over $\mathbf{X}$ for each $j = 1, \ldots, m$.

The BNs $\{\langle \mathcal{G}, \mathbb{P}_j\rangle\}_{j=1}^m$ are said to be the *compatible* BNs of the CN considered in Definition 2.

The CN $\langle \mathcal{G}, \{\mathbb{P}_1, \ldots, \mathbb{P}_m\}\rangle$ can be used to determine the following credal set:

$$K(\mathbf{X}) := \mathrm{CH}\{P_1(\mathbf{X}), \ldots, P_m(\mathbf{X})\}, \qquad (2)$$

where CH denotes the convex hull of a set of functions, and the joint mass functions $\{P_j(\mathbf{X})\}_{j=1}^m$ are those determined by the compatible BNs of the CN. With an abuse of terminology, we call the credal set in Equation (2) the *strong extension* of the CN, by analogy with the notion provided in the special case of *separately specified* CNs (see Section 4). Inference over a CN is intended as the computation of upper and lower expectations for a given function of $\mathbf{X}$ over the credal set $K(\mathbf{X})$, or equivalently over its vertices (Walley, 1991).

## 4 Separately specified credal nets

The main feature of probabilistic graphical models, which is the specification of a global model through a collection of sub-models local to the nodes of the graph, contrasts with Definition 2, which represents a CN as an explicit enumeration of BNs.

Nevertheless, there are specific subclasses of CNs that define a set of BNs as in Definition 2 through local specifications. This is for example the case of CNs with separately specified credal sets,[2] which are simply called *separately specified CNs* in the following. This specification requires each conditional mass function to belong to a (conditional) credal set, according to the following:

**Definition 3.** A separately specified CN over $\mathbf{X}$ is a pair $\langle \mathcal{G}, \mathbb{K}\rangle$, where $\mathbb{K}$ is a set of conditional credal sets $K(X_i|\pi_i)$, one for each $X_i \in \mathbf{X}$ and $\pi_i \in \Omega_{\Pi_i}$.

The *strong extension* $K(\mathbf{X})$ of a separately specified CN is defined as the convex hull of the joint mass functions $P(\mathbf{X})$, with, for each $\mathbf{x} \in \Omega_{\mathbf{X}}$:

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\pi_i), \qquad \begin{array}{l} P(X_i|\pi_i) \in K(X_i|\pi_i), \\ \text{for each } X_i \in \mathbf{X}, \pi_i \in \Pi_i. \end{array} \qquad (3)$$

Here $K(X_i|\pi_i)$ can be replaced by the set of its vertices (see Proposition 1 in the appendix). Separately specified CNs are the most popular type of CN.

As a more general case, some authors considered so-called *extensive* specifications of CNs (Ferreira da Rocha and Cozman, 2002), where instead of a separate specification for each conditional mass function as in Definition 3, the generic probability table $P(X_i|\Pi_i)$, i.e., a function of both $X_i$ and $\Pi_i$, is defined to belong to a finite set of tables. The strong extension of an extensive CN is obtained as in Equation (3), by simply replacing the separate requirements for each single conditional mass function, with extensive requirements about the tables which take values in the corresponding finite sets.

In the next section, we provide an alternative definition of CN, with the same generality of Definition 2, but obtained through local specification as in Definition 3.

[2] Some authors use also the expression *locally defined* CNs (Cozman, 2000).

## 5 Local specification of credal nets

In this section we provide an alternative and yet equivalent definition for CNs with respect to Definition 2, which is inspired by the formalism of *decision networks* (Zhang et al., 1993) via the CCM transformation (Cano et al., 1994).

**Definition 4.** A locally specified credal network over $\mathbf{X}'$ is a triplet $\langle \mathcal{G}, (\mathbf{X}_D, \mathbf{X}'), (\mathbb{O}, \mathbb{P}) \rangle$ such that: (i) $\mathcal{G}$ is a DAG over $\mathbf{X} = \mathbf{X}_D \cup \mathbf{X}'$; (ii) $\mathbb{O}$ is a collection of sets $\Omega_{X_i}^{\pi_i} \subseteq \Omega_{X_i}$, one for each $X_i \in \mathbf{X}_D$ and[3] $\pi_i \in \Pi_i$; (iii) $\mathbb{P}$ is a set of conditional mass functions $P(X_i|\pi_i)$, one for each $X_i \in \mathbf{X}'$ and $\pi_i \in \Omega_{\Pi_i}$.

We intend to show that Definition 4 specifies a CN over the variables in $\mathbf{X}'$; the nodes corresponding to $\mathbf{X}'$ are therefore called *uncertain* and will be depicted by circles, while those corresponding to $\mathbf{X}_D$ are said *decision nodes* and will be depicted by squares. Let us associate each decision node $X_i \in \mathbf{X}_D$ with a so-called *decision function* $f_{X_i} : \Omega_{\Pi_i} \to \Omega_{X_i}$ returning an element of $\Omega_{X_i}^{\pi_i}$ for each $\pi_i \in \Omega_{\Pi_i}$. Call *strategy* $\mathbf{s}$ an array of decision functions, one for each $X_i \in \mathbf{X}_D$. We denote as $\Omega_{\mathbf{S}}$ the set of all the possible strategies.

Each strategy $\mathbf{s} \in \Omega_{\mathbf{S}}$ determines a BN over $\mathbf{X}$ via Definition 4. A conditional mass function $P(X_i|\pi_i)$ for each uncertain node $X_i \in \mathbf{X}'$ and $\pi_i \in \Omega_{\Pi_i}$ is already specified by $\mathbb{P}$. To determine a BN we have then to simply represent decisions functions by mass functions: for each decision node $X_i \in \mathbf{X}_D$ and $\pi_i \in \Omega_{\Pi_i}$, we consider the conditional mass functions $P_{\mathbf{s}}(X_i|\pi_i)$ assigning all the mass to the value $f_{X_i}(\pi_i)$, where $f_{X_i}$ is the decision function corresponding to $\mathbf{s}$. The BN obtained in this way will be denoted as $\langle \mathcal{G}, \mathbb{P}_{\mathbf{s}} \rangle$, while for the corresponding joint mass function, we clearly have, for each $\mathbf{x} = (\mathbf{x}_D, \mathbf{x}') \in \Omega_{\mathbf{X}}$:

$$P_{\mathbf{s}}(\mathbf{x}_D, \mathbf{x}') = \prod_{X_l \in \mathbf{X}_D} P_{\mathbf{s}}(x_l|\pi_l) \cdot \prod_{X_i \in \mathbf{X}'} P(x_i|\pi_i). \tag{4}$$

The next step is then obvious: we want to define a CN by means of the set of BNs determined by all the possible strategies. But the point is whether or not all these networks have the same DAG, as required by Definition 2. To show this we need the following:

**Transformation 1.** *Given a locally specified CN $\langle \mathcal{G}, (\mathbf{X}_D, \mathbf{X}'), (\mathbb{P}, \mathbb{O}) \rangle$, obtain a DAG $\mathcal{G}'$ associated to the variables $\mathbf{X}'$ iterating, for each $X_d \in \mathbf{X}_D$, the following operations over $\mathcal{G}$: (i) connect with an arc all the parents of $X_d$ with all the children of $X_d$; (ii) remove the node corresponding to $X_d$.*

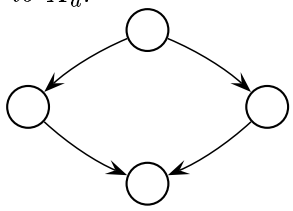


Figure 1: The DAG $\mathcal{G}'$ returned by Transformation 1 given a locally specified CN whose DAG is that in Figure 2 (or also Figure 3 or Figure 4).

Figure 1, reports an example of Transformation 1. The DAG $\mathcal{G}'$ returned by Transformation 1 is considered by the following:

**Theorem 1.** *The marginal for $\mathbf{X}'$ relative to $\langle \mathcal{G}, \mathbb{P}_{\mathbf{s}} \rangle$, i.e. the mass function $P_{\mathbf{s}}(\mathbf{X}')$ such that*

$$P'_{\mathbf{s}}(\mathbf{x}') = \sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_D}} P_{\mathbf{s}}(\mathbf{x}_D, \mathbf{x}'), \tag{5}$$

*for each $\mathbf{x}' \in \Omega_{\mathbf{X}'}$, factorizes as the joint mass function of a BN $\langle \mathcal{G}', \mathbb{P}'_{\mathbf{s}} \rangle$ over $\mathbf{X}'$, where $\mathcal{G}'$ is the DAG obtained from $\mathcal{G}$ by Transformation 1.*

From which, considering the BNs $\langle \mathcal{G}', \mathbb{P}'_{\mathbf{s}} \rangle$ for each strategy $\mathbf{s} \in \Omega_{\mathbf{S}}$, it is possible to conclude:

**Corollary 1.** *A locally specified CN as in Definition 4 properly defines a CN over $\mathbf{X}'$, based on the DAG $\mathcal{G}'$ returned by Transformation 1.*

It is worthy to note that any CN defined as in Definition 2 can be reformulated as in Definition 4, by simply adding a single decision node, which is parent of all the other nodes (see Figure 2).

The conditional mass functions corresponding to different values of the decision node are assumed to be those specified by the compatible BNs. This means that, if $D$ denotes the decision

[3] If $X_i$ corresponds to a parentless node of $\mathcal{G}$, a single set equal to the whole $\Omega_{X_i}$ is considered.

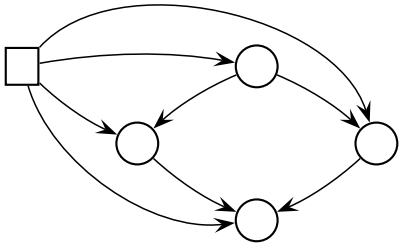


Figure 2: Local specification of a non-separately specified CN over the DAG in Figure 1. Remember that circles denote uncertain nodes, while the square is used for the decision node.

node, the states of $D$ index the compatible BNs, and $P(X_i|\pi_i, d) := P_d(X_i|\pi_i)$, where $P_d(X_i|\pi_i)$ are the conditional mass functions specified by the $d$-th compatible BN for each $X_i \in \mathbf{X}'$ and $\pi_i \in \Omega_{\Pi_i}$ and $d \in \Omega_D$. This formulation, which is an example of the CCM transformation (Cano et al., 1994), is only seemingly local, because of the arcs connecting the decision node with all the uncertain nodes. In the remaining part of this section, we show how different CNs specifications can be reformulated as required by Definition 4.

For example, we can represent extensive CNs by introducing a decision parent for each node of the original CN (Figure 3).

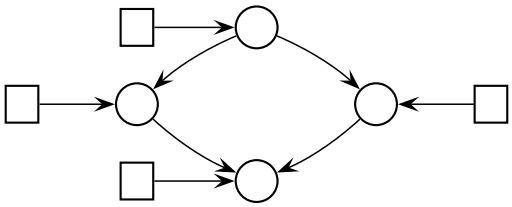


Figure 3: Local specification of an extensive CN over the DAG in Figure 1.

The conditional mass functions of the uncertain nodes corresponding to different values of the related decision nodes are assumed to be those specified by the different tables in the extensive specification of the uncertain node. This means that, if $X_i$ in an uncertain node and $D_i$ the corresponding decision node, the states $d_i \in \Omega_{D_i}$ index the tables $P_{d_i}(X_i|\Pi_i)$ of the extensive specification for $X_i$, and $P(X_i|d_i, \pi_i)$ is the joint mass function $P_{d_i}(X_i|\pi_i)$ associated to the $d_i$-th table of the extensive specification.

More generally, constraints for the specifications of conditional mass functions relative to different nodes are similarly represented by decision nodes which are the parents of these nodes.

Finally, to locally specify, as required by Definition 4, a separately specified CN, it would suffice to reformulate the separately specified CN as an extensive CN whose tables are obtained considering all the combinations of the vertices of the separately specified conditional credal sets of the same variable. Yet, this approach suffers an obvious exponential explosion of the number of tables in the input size.

A more effective procedure consists in adding a decision node in between each node and its parents, regarding the nodes of the original model as uncertain nodes (Figure 4). To complete the local specification proceed as follows. For each uncertain node $X_i$, the states $d_i \in \Omega_{D_i}$ of the corresponding decision node $D_i$ are assumed to index the vertices of *all* the conditional credal sets $K(X_i|\pi_i) \in \mathbb{K}$, with $\pi_i \in \Omega_{\Pi_i}$. In this way, for each uncertain node $X_i$, it is possible to set the conditional mass function $P(X_i|d_i)$ to be the vertex of the conditional credal set $K(X_i|\pi_i)$ associated to $d_i$, for each $d_i \in \Omega_{D_i}$. Regarding decision nodes, for each decision node $D_i$ and the related value $\pi_i$ of the parents, we simply set the subset $\Omega_{D_i}^{\pi_i} \subseteq \Omega_{D_i}$ to be such that $\{P(X_i|d_i)\}_{d_i \in \Omega_{D_i}^{\pi_i}}$ are the vertices of $K(X_i|\pi_i)$. This approach, which is clearly linear in the input size, takes inspiration from *probability trees* representations (Cano and Moral, 2002).

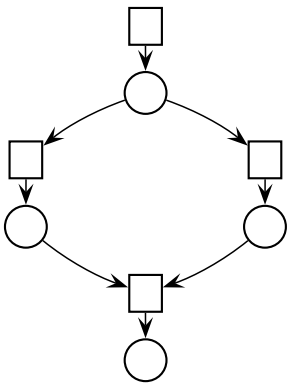


Figure 4: Local specification of a separately specified CN over the DAG in Figure 1.

Summarizing, we can obtain in efficient time a local specification of a CN, by simply considering one of the transformations described in this section.[4] It is worth noting that local specifications of CNs based on the DAGs respectively in Figure 4, Figure 3 and Figure 2 correspond to CNs based on the DAG in Figure 1 which are respectively separately specified, extensively specified and non-separately (and non-extensively) specified. Vice versa, it is easy to check that Transformation 1 can be regarded as the inverse of these three transformations.

## 6 Reasons for non-separately specified credal nets

Let us illustrate by a few examples that the necessity of non-separately specified credal nets arises naturally in a variety of problems.

**Conservative Inference Rule** The *conservative inference rule* (CIR) is a new rule for updating beliefs with incomplete observations (Zaffalon, 2005). CIR models the case of mixed knowledge about the incompleteness process, which is assumed to be nearly unknown for some variables and unselective for the others. This leads to an imprecise probability model, where all the possible completions of the incomplete observations of the first type of variables are considered. In a recent work, updating beliefs with CIR has been considered for BNs (Antonucci and Zaffalon, 2006). The problem is proved to be equivalent to a traditional belief updating problem in a CN: the equivalence is made possible by non-separately specified credal nets.

**Qualitative networks** Qualitative probabilistic networks (Wellman, 1990) are an abstraction of BNs, where the probabilistic assessments are replaced by qualitative relations of the influences or the synergies between the variables. If we regard qualitative nets as credal nets, we see that not all types of relations can be represented by separate specifications of the conditional credal sets. This is, for instance, the case of (positive) *qualitative influence*, which requires, for two boolean variables $A$ and $B$, that

$$P(a|b) \geq P(a|\neg b). \tag{6}$$

The qualitative influence between $A$ and $B$ can therefore be modeled by requiring $P(A|b)$ and $P(A|\neg b)$ to belong to credal sets, which cannot be separately specified because of the constraint in Equation (6). An extensive specification for $A$ should therefore be considered to model the positive influence of $B$ (Cozman et al., 2004).

**Equivalent graphs for CNs** Remember that DAGs represent independencies between variables according to the Markov condition. Different DAGs describing the same independencies are said to be *equivalent* (Verma and Pearl, 1990). Thus, a BN can be reformulated using an equivalent DAG. The same holds with CNs, when (as implicitly done in this paper) *strong independence* replaces standard probabilistic independence in the Markov condition (Moral and Cano, 2002).

Consider, for example, $A \rightarrow B$ and $A \leftarrow B$, which are clearly equivalent DAGs. One problem with separately specified CNs is that they are not closed under this kind of (equivalent) structure changes: if we define a separately specified CN for $A \rightarrow B$, and then reverse the arc, the resulting net will not be separately specified in general. Consider the following separate specifications of the conditional credal sets for a CN over $A \rightarrow B$:

$$\begin{aligned} K(B|a) &:= [.9,.1] \quad K(B|\neg a) := [.8,.2] \\ K(A) &:= \mathrm{CH}\{[.7,.3],[.5,.5]\}, \end{aligned}$$

where two-dimensional horizontal arrays are used to denote mass functions for boolean variables. Such a separately specified CN has two compatible BNs, one for each vertex of $K(A)$. From the joint mass functions corresponding to

[4] As a side note, it is important to be aware that a credal set can have a very large number of vertices, and this can still be a source of computational problems for algorithms (such as those based on the CCM transformation) that explicitly enumerate the vertices of a net's credal sets. This is a well-know issue, which in the present setup is related to the possibly large number of states for the decision nodes in the locally specified representation of a credal net.

these BNs, say $P_1(A,B)$ and $P_2(A,B)$, we obtain the conditional mass functions for the corresponding BNs over $B \to A$:

$$P_1(B) = [.87, .13] \qquad P_2(B) = [.85, .15]$$
$$P_1(A|b) = [.72, .28] \qquad P_2(A|b) = [.52, .47]$$
$$P_1(A|\neg b) = [.54, .46] \qquad P_2(A|\neg b) = [.33, .67].$$

According to Definition 2, these two distinct specifications define a CN over $B \to A$, which cannot be separately specified as in Definition 3. To see this, note for example that the specification $P(a|b) = .52$, $P(a|\neg b) = .33$ and $P(b) = .85$, which is clearly a possible specification if the conditional credal sets were separately specified, would lead to the unacceptable mass function $P(A) = [.49, .51] \notin K(A)$.

It is useful to observe that general, non-separately specified, CNs do not suffer for these problems just because they are closed under equivalent changes in the structure of the DAG.

**Learning from incomplete data** Given three boolean random variables $A$, $B$ and $C$, let the DAG $A \to B \to C$ express independencies between them. We want to learn the model probabilities for such a DAG from the incomplete data set in Table 1, assuming no information about the process making the observation of $B$ missing in the last record of the data set. The most conservative approach is therefore to learn two distinct BNs from the two complete data sets corresponding to the possible values of the missing observation and consider indeed the CN made of these compatible BNs.

| A | B | C |
|---|---|---|
| $a$ | $b$ | $c$ |
| $\neg a$ | $\neg b$ | $c$ |
| $a$ | $b$ | $\neg c$ |
| $a$ | $*$ | $c$ |

Table 1: A data set about three boolean variables, $*$ denotes a missing observation.

To make things simple we compute the probabilities for the joint states by means of the relative frequencies in the complete data sets. Let $P_1(A,B,C)$ and $P_2(A,B,C)$ be the joint mass function obtained in this way, which define the same conditional mass functions for:

$$P_1(A) = P_2(A) = [.75, .25]$$
$$P_1(B|\neg a) = P_2(B|\neg a) = [0, 1]$$
$$P_1(C|\neg b) = P_2(C|\neg b) = [1, 0];$$

and different conditional mass functions for:

$$P_1(B|a) = [1, 0] \qquad P_2(B|a) = [.67, .33]$$
$$P_1(C|b) = [.67, .33] \qquad P_2(C|b) = [.5, .5].$$

We have therefore obtained two BNs over $A \to B \to C$, which can be regarded as the compatible BNs of a CN. Such a CN is clearly non-separately specified, because the two BNs specify different conditional mass functions for more than a variable.

## 7 From locally to separately specified credal nets

In this section, we prove that any locally specified CN over $\mathbf{X}'$ can equivalently be regarded as a separately specified CN over $\mathbf{X}$. The transformation is technically straightforward: it is based on representing decision nodes by uncertain nodes with vacuous conditional credal sets, as formalized below.

**Transformation 2.** *Given a locally specified CN $\langle \mathcal{G}, (\mathbf{X}_D, \mathbf{X}'), (\mathbb{O}, \mathbb{P}) \rangle$ over $\mathbf{X}'$, obtain a separately specified CN $\langle \mathcal{G}, \mathbb{K} \rangle$ over $\mathbf{X}$, where the conditional credal sets in $\mathbb{K}$ are for each $X_i \in \mathbf{X}$ and $\pi_i \in \Omega_{\Pi_i}$:*

$$K(X_i|\pi_i) := \begin{cases} P(X_i|\pi_i) & \text{if } X_i \in \mathbf{X}' \\ K_{\Omega^{\pi_i}_{X_i}}(X_i) & \text{if } X_i \in \mathbf{X}_D, \end{cases} \tag{7}$$

*where $P(X_i|\pi_i)$ is the mass function specified in $\mathbb{P}$ and $K_{\Omega^{\pi_i}_{X_i}}(X_i)$ the* vacuous *credal set for $\Omega^{\pi_i}_{X_i}$.*

Figure 5 reports an example of Transformation 2, which is clearly linear in the input size.

The (strong) relation between a locally specified CN $\langle \mathcal{G}, (\mathbf{X}_D, \mathbf{X}'), (\mathbb{O}, \mathbb{P}) \rangle$ over $\mathbf{X}'$ and the separately specified CN $\langle \mathcal{G}, \mathbb{K} \rangle$ returned by Transformation 2 is outlined by the following:

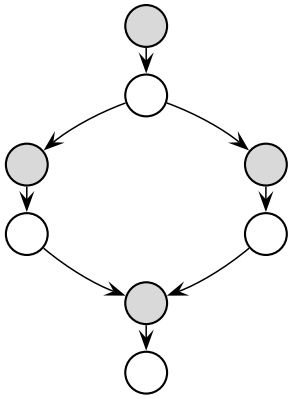


Figure 5: The DAG associated to the separately specified CN returned by Transformation 2, from the locally specified CN based on the DAG in Fig. 4. The conditional credal sets of the white nodes (corresponding to the original uncertain nodes) are precisely specified, while the grey nodes (i.e., new uncertain nodes corresponding to the former decision nodes) represent variables whose conditional credal sets are vacuous.

**Theorem 2.** *Let $\tilde{K}(\mathbf{X}')$ be the marginal for $\mathbf{X}'$ of the strong extension $\tilde{K}(\mathbf{X})$ of $\langle \mathcal{G}, \mathbb{K} \rangle$ and $K(\mathbf{X}')$ the strong extension of $\langle \mathcal{G}, (\mathbf{X}_D, \mathbf{X}'), (\mathbb{O}, \mathbb{P}) \rangle$. Then:*

$$K(\mathbf{X}') = \tilde{K}(\mathbf{X}'). \tag{8}$$

From Theorem 2, it is straightforward to conclude the following:

**Corollary 2.** *Any inference problem on a locally specified CN can be equivalently solved in the separately specified CN returned by Transformation 2.*

Let us stress that Transformation 2 is very simple, and it is surprising that it is presented here for the first time, as it is really the key to "separate" the credal sets of non-separately specified nets: in fact, given a non-separately specified CN, one can locally specify the CN, using the prescriptions of the second part of Section 5, and apply Transformation 2 to obtain a separately specified CN. According to Corollary 2, then, any inference problem on the original CN can equivalently be represented on this new separately specified CN. To make an example, this procedure can immediately solve the CIR updating problem described in Section 6, which is a notable and ready-to-use result.

## 8 Conclusions and outlooks

We have defined a new graphical language to formulate any type of credal network, both separately and non-separately specified. We have also showed that any net represented with the new language can be easily transformed into an equivalent separately specified credal net. This implies, in particular, that non-separately specified nets have an equivalent separately specified representation, for which solutions algorithms are available in the literature.

The transformation proposed also shows that a subclass of separately specified credal networks can be used to solve inference problems for arbitrary specified credal nets: this is the class of nets in which the credal sets are either vacuous or precise. It is worth noting that a recent development of the approximate L2U algorithm (Antonucci et al., 2006) seems to be particularly suited just for such a class, and should therefore be considered in future work.

Finally, the strong connection between the language for credal networks introduced in this paper and the formalism of decision networks (including influence diagrams), seems to be particularly worth exploring for cross-fertilization between the two fields.

## Acknowledgements

This research was partially supported by the Swiss NSF grant 200020-109295/1.

## A Proofs

The obvious proofs of Corollary 1 and Corollary 2 are omitted.

*Proof of Theorem 1.* Let us start the marginalization in Equation (5) from a decision node $X_j \in \mathbf{X}_D$. According to Equation (4), for each $\mathbf{x} \in \Omega_{\mathbf{X}}$:

$$\sum_{\mathbf{x}_j \in \Omega_{\mathbf{X}_j}} P_{\mathbf{s}}(\mathbf{x}) = \sum_{\mathbf{x}_j \in \Omega_{\mathbf{X}_j}} \left[ \prod_{X_l \in \mathbf{X}_D} P_{\mathbf{s}}(x_l|\pi_l) \cdot \prod_{X_i \in \mathbf{X}'} P(x_i|\pi_i) \right]. \tag{9}$$

Thus, moving out of the sum the conditional probabilities which do not refer to the states of $X_j$ (which are briefly denoted by $\Delta$), Equation (9) becomes:

$$\Delta \cdot \sum_{x_j \in \Omega_{X_j}} \left[ P_{\mathbf{s}}(x_j|\pi_j) \cdot \prod_{X_r \in \Gamma_{X_j}} P(x_r|x_j, \tilde{\pi}_r) \right], \tag{10}$$

where $\Gamma_{X_j}$ denotes the children of $X_j$ and, for each $X_r \in \Gamma_{X_j}$, $\tilde{\Pi}_r$ are the parents of $X_r$ deprived of $X_j$. Therefore, considering that the mass function $P_{\mathbf{s}}(X_j|\pi_j)$ assigns all the mass to the value $f_{X_j}(\pi_j) \in \Omega_{X_j}$, where $f_{X_j}$ is the decision function associated to $\mathbf{s}$, Equation (10) rewrites as

$$\Delta \cdot \prod_{X_r \in \Gamma_{X_j}} P(x_r|f_{X_j}(\pi_j), \tilde{\pi}_r). \tag{11}$$

It is therefore sufficient to set $\Pi'_r := \Pi_j \cup \tilde{\Pi}_r$, and

$$P'_{\mathbf{s}}(X_r|\pi'_r) := P(X_r|f_{X_j}(\pi_j), \tilde{\pi}_r), \tag{12}$$

to regard Equation (11) as the joint mass function of a BN over $\mathbf{X} \setminus \{X_j\}$ based on the DAG returned by Transformation 1 considered for the single decision node $X_j \in \mathbf{X}_D$. The thesis therefore follows from a simple iteration over all the $X_j \in \mathbf{X}_D$. □

The following well-known and (relatively) intuitive proposition is required to obtain Theorem 2, and will be proved here because of the seemingly lack of its formal proof in the literature.

**Proposition 1.** *The vertices $\{\tilde{P}_j(\mathbf{X})\}_{j=1}^m$ of the strong extension $\tilde{K}(\mathbf{X})$ of a separately specified CN $\langle \mathcal{G}, \mathbb{K} \rangle$ are joint mass functions obtained by the combination of vertices of the separately specified conditional credal sets, i.e., for each $\mathbf{x} \in \Omega_{\mathbf{X}}$:*

$$\tilde{P}_j(\mathbf{x}) = \prod_{i=1}^{n} \tilde{P}_j(x_i|\pi_i), \tag{13}$$

*for each $j = 1, \ldots, m$, where, for each $i = 1, \ldots, n$ and $\pi_i \in \Omega_{\Pi_i}$, $\tilde{P}_j(X_i|\pi_i)$ is a vertex of $K(X_i|\pi_i) \in \mathbb{K}$.*

*Proof of Proposition 1.* We prove the proposition by a *reductio ad absurdum*, assuming that at least a vertex $\tilde{P}(\mathbf{X})$ of $\tilde{K}(\mathbf{X})$ is not obtained by a local combination of vertices of the conditional credal sets in $\mathbb{K}$. This means that, for each $\mathbf{x} \in \Omega_{\mathbf{X}}$, $\tilde{P}(\mathbf{x})$ factorizes as in Equation (13), but at least a conditional probability in this product comes from a conditional mass function which is not a vertex of the relative conditional credal set. This conditional mass function, say $P(X_t|\pi_t)$, can be expressed as a convex combination of vertices of $K(X_t|\pi_t)$, i.e., $P(X_t|\pi_t) = \sum_\alpha c_\alpha P_\alpha(X_t|\pi_t)$, with $\sum_\alpha c_\alpha = 1$ and, for each $\alpha$, $c_\alpha \geq 0$ and $P_\alpha(X_t|\pi_t)$ is a vertex of $K(X_t|\pi_t)$. Thus, for each $\mathbf{x} \in \Omega_{\mathbf{X}}$,

$$\tilde{P}(\mathbf{x}) = \left[ \sum_\alpha c_\alpha P_\alpha(x_t|\pi_t) \right] \cdot \prod_{i \neq t} P(x_t|\pi_t), \tag{14}$$

which can be easily reformulated as a convex combination. Thus, $\tilde{P}(\mathbf{X})$ is a convex combination of elements of the strong extension $\tilde{K}(\mathbf{X})$. This violates the assumption that $\tilde{P}(\mathbf{X})$ is a vertex of $\tilde{K}(\mathbf{X})$. □

*Proof of Theorem 2.* According to Theorem 1, the strong extension $K(\mathbf{X}')$ of $\langle \mathcal{G}, (\mathbf{X}_D, \mathbf{X}'), (\mathbb{O}, \mathbb{P}) \rangle$ can be regarded as the marginal for $\mathbf{X}'$ of

$$K(\mathbf{X}) = \mathrm{CH}\{P_{\mathbf{s}}(\mathbf{X})\}_{\mathbf{s} \in \Omega_{\mathbf{S}}}, \tag{15}$$

where for each $\mathbf{s} \in \Omega_{\mathbf{S}}$, $P_{\mathbf{s}}(\mathbf{X})$ is the joint mass function associated to $\langle \mathcal{G}, \mathbb{P}_{\mathbf{s}} \rangle$. For each $X_d \in \mathbf{X}_D$, the conditional mass functions $P_{\mathbf{s}}(X_d|\pi_d)$, specified for each $\pi_d \in \Omega_{\Pi_d}$ in $\mathbb{P}_{\mathbf{s}}$, assign all the mass to the single state $f_{X_d}(\pi_d) \in \Omega_{X_d}^{\pi_d}$, where $f_{X_d}^{\pi_d}$ are the decision functions associated to the strategy $\mathbf{s} \in \Omega_{\mathbf{S}}$, and represent therefore a vertex of the vacuous conditional credal set $K_{\Omega_{X_d}}^{\pi_d}(X_d) \in \mathbb{K}$ specified in Equation (7). Thus, $P_{\mathbf{s}}(\mathbf{X})$ is a vertex of $\tilde{K}(\mathbf{X})$ because of Proposition 1. As $\mathbf{s}$ varies in $\Omega_{\mathbf{S}}$, all the vertices of $\tilde{K}(\mathbf{X})$ are obtained and therefore $\tilde{K}(\mathbf{X}) = K(\mathbf{X})$, from which the thesis follows marginalizing. □

## References


A. Antonucci and M. Zaffalon. 2006. Equivalence between Bayesian and credal nets on an updating problem. In *Proceedings of third international conference on Soft Methods in Probability and Statistics*. To appear.

A. Antonucci, M. Zaffalon, J.S. Ide, and F.G. Cozman. 2006. Binarization algorithms for approximate updating in credal nets. In *Proceedings of the third European Starting AI Researcher Symposium*. To appear.

A. Cano and S. Moral. 2002. Using probability trees to compute marginals with imprecise probabilities. *Int. J. Approx. Reasoning*, 29(1):1–46.

A. Cano, J. Cano, and S. Moral. 1994. Convex sets of probabilities propagation by simulated annealing on a tree of cliques. In *Proceedings of 5th International Conference on Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 4–8.

F. G. Cozman, C. P. de Campos, J. S. Ide, and J. C. Ferreira da Rocha. 2004. Propositional and relational Bayesian networks associated with imprecise and qualitative probabilistic assessments. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence*, pages 104–111. AUAI Press.

F. G. Cozman. 2000. Credal networks. *Artificial Intelligence*, 120:199–233.

F.G. Cozman. 2005. Graphical models for imprecise probabilities. *Int. J. Approx. Reasoning*, 39(2-3):167–184.

J. C. Ferreira da Rocha and F. G. Cozman. 2002. Inference with separately specified sets of probabilities in credal networks. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 430–437. Morgan Kaufmann.

I. Levi. 1980. *The Enterprise of Knowledge*. MIT Press, London.

S. Moral and A. Cano. 2002. Strong conditional independence for credal sets. *Annals of Mathematics and Artificial Intelligence*, 35(1-4):295–321.

T. Verma and J. Pearl. 1990. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pages 255–270. Elsevier.

P. Walley. 1991. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, New York.

M. P. Wellman. 1990. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44(3):257–303.

M. Zaffalon. 2001. Statistical inference of the naive credal classifier. In *Proceedings of the Second International Symposium on Imprecise Probabilities and Their Applications*, pages 384–393. Shaker.

M. Zaffalon. 2005. Conservative rules for predictive inference with incomplete data. In *Proceedings of the Fourth International Symposium on Imprecise Probabilities and Their Applications*, pages 406–415. SIPTA.

N. Zhang, R. Qi, and D. Poole. 1993. A computational theory of decision networks. *Int. J. Approx. Reasoning*, 11(2):83–158.

# A Bayesian Network Framework for the Construction of Virtual Agents with Human-like Behaviour

Olav Bangsø, Nicolaj Søndberg-Madsen and Finn Verner Jensen
{bangsy, nicolai, fvj}@cs.aau.dk
Department of Computer Science
Aalborg University, Denmark

**Abstract**

It seems tempting to equip virtual agents, for example in a computer game, with graphical models for decision making. That is, if a virtual agent possesses an influence diagram of its relevant world, then it will be able to take rational decisions and appear very intelligent. However, this approach misses the point that agents are supposed to act human-like regardless how alien they may appear. What is missing is that emotions influence the choice of action, and rather than trying to incorporate emotions in the utilities it is preferable to make explicit the dynamics of emotions as well as their influence on the choice. The Twilight project performed at Aalborg University is based on the so called EMS, which is a system for simulating human emotions. According to this theory, emotions are affected when events influence goals or resources. The Comparator analyses how an event may influence goals and resources. The Evaluator determines the change of emotional profile (a vector containing degree of anger, fear, joy, and sorrow), and the Action Proposer picks the action mediating possibilities and emotional profile. This paper is inspired by the EMS. We focus on the Comparator and outline how the developed Bayesian network models also can be utilized for the Evaluator and the Action Proposer.

## 1 Introduction

Virtual agents acting credibly as human beings would be of interest in computer games. The class of games that would benefit the most from virtual agents acting credibly as humans is *Role-Playing Games* (RPGs). RPGs usually give the player more freedom than other classes of games. With this freedom the player can be expected to interact with any object in the game, including any virtual agent. Games are getting more and more realistic in all aspects however in many cases little effort is put into the virtual agents. This in spite of computer agents acting credibly as humans is among the oldest investigated topics in AI (Turing, 1950). We present a method to produce agents with human-like behaviour which is well suited for virtual agents in RPGs.

Some popular games, which have very simple virtual agents, are Diablo II (Blizzard Entertainment, 2000) and Neverwinter Nights (Bioware, 2002). In Diablo II each agent exhibit the same behaviour except from certain scripted events. In Neverwinter Nights the agents do not have independent behaviour; they wait only for the player to come and interact with them. Usually, rational virtual agents attempt to maximize their expected utility. This solution, however, results in predictable behaviour which usually is unwanted in computer games.

One of the aims of the Twilight project is to develop a system for simulating human emotions, the system is called an *Emotional and Motivational System* (EMS). Other aims include a system for dynamic narration and modeling of the players preferences based on the players action. Nikolaj Hyldig from the *Department of Communication and Psychology, Aalborg University* has proposed to base the system on theories on the human emotional and motivational system (i.e. psychology) which guides human

actions (Hyldig, 2005).

The EMS is similar to the OCC model (Ortony, Clore, and Collins, 1988). In (Bartneck, 2002) some practical problems with implementing the OCC model for facial expressions are pointed out, along with solutions for some of them. Most notable are a reduction of the number of emotions and keeping a history of events. Expressing several emotions at once remains a problem along with modeling different personality types.

In this work we focus on actions and utilize *Bayesian Networks* (BNs) to simulate emotional responses. The driving force in the EMS is *Quality of life* (QoL) and it is used to determine both emotional state and choice of action. The choice of action is influenced by the value of all emotions, allowing several emotions to be expressed through the agents' actions. Using the resources available to the agent to model the world state for the agent, we eliminate the need for the history function proposed by (Bartneck, 2002). By allowing predefined preferences to influence both the change in emotion and choice of action it is possible to model different personality types

Each agent will get a certain amount of QoL by performing activities. Each activity requires a number of resources in order to be carried out and if a resource is lost it affects which activities the agent can carry out. The QoL received from a certain activity is determined not only by how effective the activity is but also by how motivated the agent is and by his general preference for that activity. We use QoL to determine the *Emotional Profile* ($\overrightarrow{EP}$) of the virtual agents.

We work with four kinds of emotions: *Joy*, *Sorrow*, *Anger* and *Fear*. The exact emotional response to an event depends on the change in the level of expected QoL in the situation after the event. *Anger* occurs when the agent estimates that it is difficult to retain the level of QoL it had before the event while *Sorrow* occurs when it has become unachievable. *Fear* occurs when the agent estimates a significant decrease in future expected QoL while *Joy* occurs when the expected QoL rises. The intensity of the emotion is determined by the change in expected QoL.

The EMS is decomposed into five steps. The *Comparator* compares how likely it is that the agent can retain or maintain its QoL before and after an event has occured. The *Evaluator* determines the emotional reaction to the new situation. It also determines the intensity of the emotional response. The *Action Proposer* proposes the action the agent will take in order to deal with the event. The action chosen should be influenced by the expected QoL and the agent's $\overrightarrow{EP}$. The *Physiological Change Generator* handles changes in the agent's appearance in the game to exhibit the emotions i.e. textures, animations etc. The *Actor* carries out the chosen action, if any.

In this paper we will concentrate on the Comparator and describe how the problem can be solved using Bayesian networks. The next section will describe the general structure of the models through an example. Section 3 will show how the models can support the Comparator. The last two sections outlines future work, Section 4 will outline how the developed Bayesian network models can be utilized for the Evaluator and the Action Proposer, while Section 5 describes what must be specified for our framework to be utilized.

## 2 Virtual Agents

The goal of this work is to have virtual agents performing close to how humans would. To this end we use QoL as the basic concept driving the agents. Activities and ownership are the only ways agents can receive QoL and the QoL an agent receives from a specific activity is influenced by how motivated the agent is for doing the activity. The motivation for the different activities may be influenced by different factors, called resources, and the motivation for each of the activities is influenced by the agent's preference for the specific activity. A rational agent would try to maximize the expected QoL, but in this work the $\overrightarrow{EP}$ of the agent also impacts the choice of activity.

The methodology for deciding which activity to perform can be described by the following

steps:

1. An event changes the resources of the agent.

2. The expected QoL may change as an effect of the change in resources.

3. A change in expected QoL triggers an emotional response.

4. The expected QoL given the changed resources and the new $\overrightarrow{EP}$ dictate the activity to perform.

We will introduce the framework through an example. We use Object Oriented Bayesian Networks (Bangsø and Wuillemin, 2000) (OOBNs) in the example for ease of exposition, as the irrelevant details are hidden in the objects, we also believe that some classes can be reused, so some gain in specification time should be achieved. However the ideas in this paper are equally applicable to BNs. Some of the notation of OOBNs are illustrated in Figure 1. The dashed nodes are input nodes representing some node specified outside the scope of the class having an influence on nodes inside instances of the class. The boxes represent instances of classes, where the nodes on the upper half of the box are input nodes to the instance, and the nodes at the bottom are output nodes; nodes inside the instance that can be parents of nodes in the surrounding class.

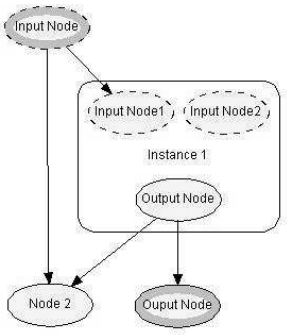



Figure 1: An OOBN containing an instance with two input nodes and an output node. The OOBN class itself has one input node and one output node.

As this work is in the context of computer games, the main focus is on having the agents exhibiting human-like behaviour, observable by the player. The agents in the example will have three types of activities that yield QoL, in a commercial game there will need to be more:

- Farming activities has to do with farming the land.
- Home activities are the activities we want the agents to be able to perform within the confines of their home.
- Spare-time activities are any activities we wish the agent to be doing when it is not working or at home.

The QoL received from each activity type is influenced by the agent's type, e.g. a soldier will not get much QoL from doing farming activities whereas it will be the meaning of existence for a farmer.

### 2.1 Farming Activities

The motivation for all farming activities are influenced by the season and the time of day. The farming activities are:

- Ploughing with a plough and a horse. The most efficient way of ploughing. Requires a field, a plough, and a horse.
- Ploughing with a plough. If no horse is available, ploughing with a plough is the most efficient. Requires a field and a plough.
- Ploughing with a shovel. The least efficient way of ploughing. Requires a field and a shovel.

The QoL received from each farming activity is influenced by the agent's energy level, e.g. if the agent has a low energy level, it is not expedient to plough with a shovel.

### 2.2 Home Activities

The home activities are:

- Cleaning. The motivation for cleaning is affected by the time of day and the season.

- Playing with children. Requires at least one child. The motivation is influenced by the agent's energy level. The QoL received is influenced by the time of day and the agent's charisma.
- Playing with spouse. Requires a spouse. The motivation is influenced by the season, the agent's energy level and the spouses charisma. The QoL is influenced by the time of day and the agent's charisma.

### 2.3 Spare-time Activities

The spare-time activities are:

- Pub visit. The motivation for going to the pub is influenced by the agent's marital status and energy level. The QoL is influenced by the time of day, whether the agent has company, and how much money the agent has.
- Dating. Requires a date. The motivation for dating is influenced by the agent's marital status and energy level. The QoL is influenced by the time of day, the marital status, the agent's charisma, and how much money the agent has.
- Playing chess. Requires a chess partner. The motivation is influenced by the time of day. The QoL is influenced by the agent's intelligence.

In Figure 2 an OOBN for the spare-time activity type can be seen. The general structure for activity type OOBNs can be extracted from this example. The parameters for the models are elicited from the game designer, as they tend to have an urge to have control over how characters in their games behave. Furthermore there will in general be no data available for the behaviour of game characters, the desired behaviour of them need not be the same as that of any population in existence.

There is an instance for each activity outputting a measure of the QoL of that activity. Each activity also has an instance outputting the agent's motivation for the specific activity. The preference for the activity type is a parent of the utility node. The input nodes are the resources, the decision node is a dummy variable and is used for reading the expected QoL of each activity.

### 2.4 Supporting the EMS

The developed OOBN models support the EMS in three ways; The OOBNs are models of expected QoL; we can enter events to the models to get the resultant change of expected QoL (part of the task of the Comparator of the EMS). The EMS should output an updated $\overrightarrow{EP}$, so it needs to determine the strength of the emotional response to the event (part of the task of the Evaluator); part of this task will include the size of the change in expected QoL. The OOBN models can provide a ranked list of candidate activities which is used in conjunction with $\overrightarrow{EP}$ to choose the activity to perform (the task of the Action Proposer).

In the following sections we describe in more detail how the models support the EMS.

## 3 Using the Models for the Comparator

Part of the EMS is the Comparator that compares the agent's expected QoL before and after an event has occured. In this section we will show how the OOBN models can be used to this end.

Any event can be modeled by how it influences the agent's resources, e.g. a theft event results in the stolen resources not being available for the agent. Once an event is modeled it can be inserted into the activity type OOBNs and the QoL yielded by the best activity given $\overrightarrow{EP}$ can be compared to the QoL yielded before the event was introduced. This gives the change in expected QoL for the event. Note that each resource also has a QoL attached for ownership, i.e. the agent gets QoL for having the resource, and if the resource is lost this will also affect the change in expected QoL. The emotional response then depends on this change as outlined in Section 1.

In Figure 3 an OOBN for the event that the agent's money is stolen can be seen. The output node *Money* should contain the expected

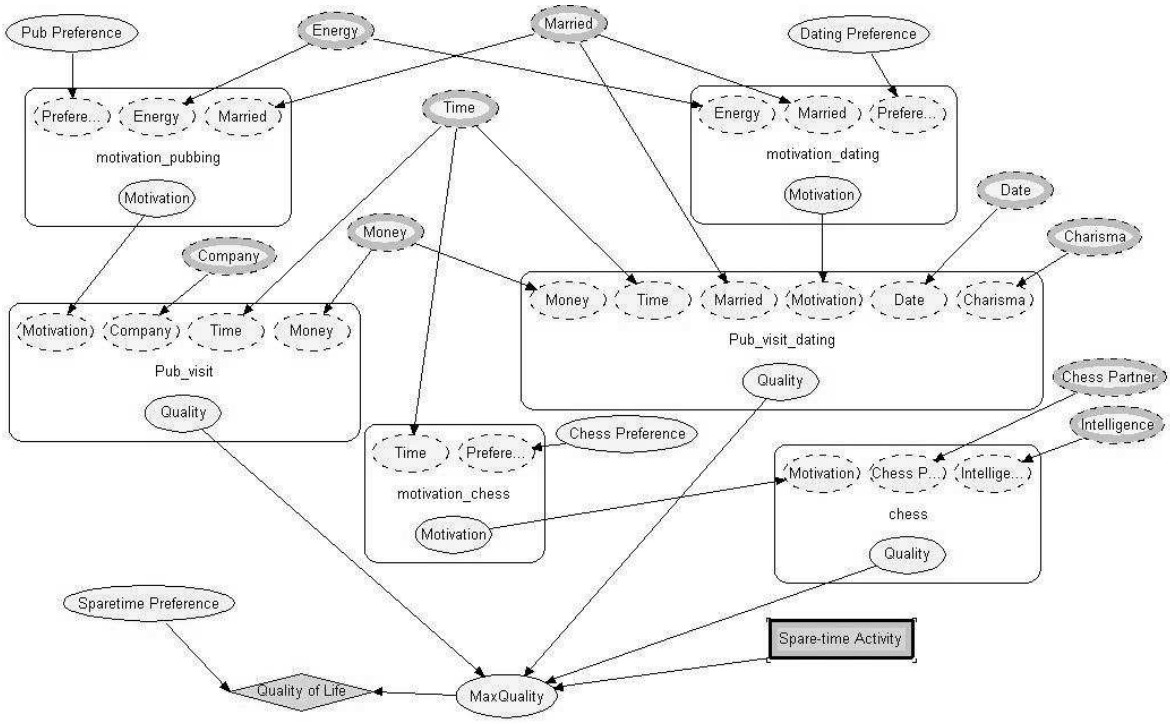



Figure 2: An object oriented BN for the spare-time activities.

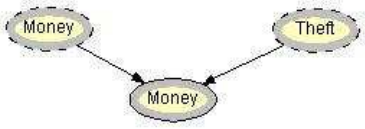



Figure 3: A model for a theft event (theft of the agent's money). The *Theft* node models whether the event has occured or if it is a threat.

value for the agent's resource money, given the current value and the probability of the event occurring. The input node *Money* should be associated with the agents' *Money* node (from the activity models). The node *Theft* contains the probability that the event will occur, i.e. the probability that the threat will be carried out. If this *Theft* node is in the state false, the output node *Money* will have the same distribution as the input node *Money*, i.e. no theft has occurred, so the agent's money has not been affected. If it is in the state true, the event has occured so the agent's money will be affected by the event, and the output *Money* will then contain the new value for the agent's money. This model is inserted in the activity type OOBNs between the Money input node and the activities where Money is an input node. This is shown in Figure 4.

## 4 Outline of the Evaluator and the Action Proposer

In this section we outline ideas concerning the Evaluator and the Action Proposer, these are still preliminary, and should be seen as topics for future research.

The Evaluator determines to which extent the four emotions are influenced by the event. Determining the emotional influence is done according to some simple rules using the change in expected QoL as input:

1. Positive change in expected QoL results in joy.

2. Negative change in expected QoL, the change cannot easily be rectified (e.g. the murder of a loved one), results in sorrow.

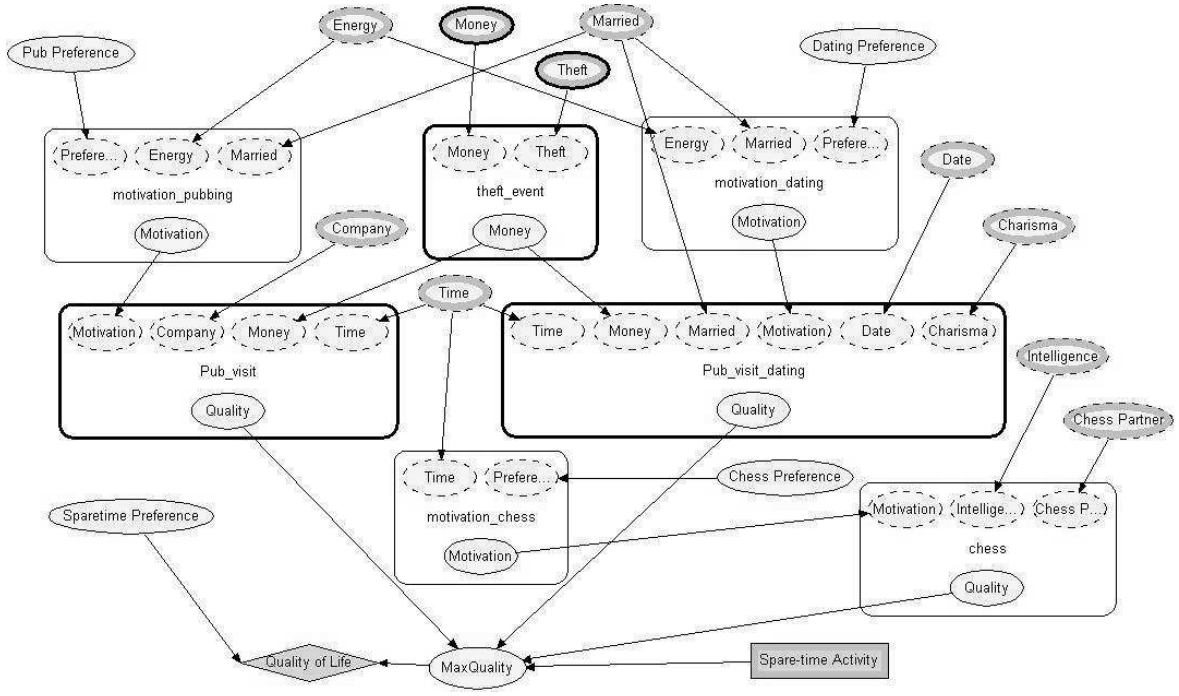



Figure 4: The Spare-time activity type OOBN where a model for a theft event (theft of the agent's money) has been inserted.

3. Negative change in expected QoL, the change can be rectified without a lot of effort, results in anger.

4. Negative change in expected QoL as a result of a threat, the antagonist can be defeated, results in anger.

5. Negative change in expected QoL as a result of a threat, the antagonist is next to impossible to defeat, results in fear.

Distinguishing between situations where 2. or 3. is fired is not a trivial matter. An approximation can be achieved by making BN models of the difficulty of getting resources, and then determining how difficult it is to bring the expected QoL back to at least what it was before the event. These models will resemble the models already described. The differences will be that activities are plans for acquiring resources, the activity types are different plans for acquiring a resource, and instead of expected QoL the utilities will be a measure of the expected inconvenience. The plans will also include an instance of a class outputting the probability of success of the plan, used to determine the expected effect on the resources of attempting the plan and for assessing the difficulty.

This approximation means that we risk ignoring the murder of a wife, unless the QoL for owning one is set high enough, as the highest expected QoL may not include a wife. Distinguishing between 4. and 5. can be done by comparing the resources of the agent and the resources of the antagonist.

Determining the strength of the influence on the emotions is a mixture of the difficulty of overcoming the problem and how serious the expected QoL is affected, i.e. how large the change in expected QoL is. Finding the strength is done during the determination of the emotion affected, the seriousness is the output of the Comparator.

The Action Proposer should output the next thing for the agent to do. It gets as input an $\overrightarrow{EP}$ of the agent and a list of activities, ranked by their expected QoL. Different agents will react differently to an emotional change, e.g. as

a bully type agent is angered it is more likely to choose aggressive activities, while a pacifist agent will react to higher anger by being more likely to choose activities where the agent is alone. We envision a system where each agent has an *Emotional Factor vector*($\overrightarrow{EF}$) with an entrance for each emotion associated with each activity. The Action Proposer will then for each activity, multiply the dot-product of $\overrightarrow{EP}$ and $\overrightarrow{EF}$ with the expected QoL yielding the *Emotional Value*($EV$):

$$EV = QoL * \overrightarrow{EP} \cdot \overrightarrow{EF}$$

To avoid predictability we will pick an activity at random using these values as weights.

Using only this will result in the agents always performing some activity, thus never trying to better their situation by trying to acquire more resources. As we already need models to assess the difficulty of (re)acquiring resources for the Evaluator, we might as well use these models for determining whether the agents should try to acquire resources. The expected inconvenience of each plan can be subtracted from the expected QoL of the situation where the plan has been attempted. This value is then used for the calculation of the $EV$ of the plan. Plans for acquiring resources can then be used by the Action Proposer in the same way as activities. Note that this approach is myopic, but can easily be expanded to include any number of subsequent plans. This, however, comes at the cost of an exponential growth in the number of possible combination of plans to evaluate. Alternatively a greedy approach may be attempted, where the first step is to find the plan, or plans, with the highest expected QoL (after subtracting the expected inconvenience of the plan), then the expected situation after carrying out the plan is used to find the best plan to execute next, and so on, until no gain in expected QoL is possible. The greedy approach will have the advantage of giving a better QoL estimate, but will take longer to calculate, for each step in the sequence, each possible next step needs to be evaluated. Furthermore this approach will include only one choice of resource acquisition[1]; the best sequence of plans, so once the $EV$ is calculated, we cannot be sure that this was in fact the best sequence.

To avoid the value of emotions being constantly growing, we will maintain a list of the contributions to $\overrightarrow{EP}$, and let the values of each contribution fade over time. Some events may, however, be so powerful that they are never completely removed from $\overrightarrow{EP}$ (traumas), how this occurs in humans is a matter of psychology, and it is being investigated in the Twilight project. Current events may also trigger the recollection of previous events, giving them renewed strength in $\overrightarrow{EP}$, this is, once again, a thing that we hope to look further into how to include.

## 5 Specification Tasks

In this section we will go through all the things that need to be specified for the outlined approach to be implemented completely. We aim at developing a system for automatically generating OOBN models from the specifications.

### 5.1 Resources

The resources that might be available for the agents and their possible values. Some resources are global values, e.g. the season, others are attributes of the agents, e.g. charisma, while others are objects or abstractions that can be found in the game-world, e.g. a plough is an object while money might be an abstraction. No matter what they are, we need to know were to get the values from the game. The QoL of each resource being in a specific state must also be specified, along with a standard preference for the resource. It may also be expedient to include the standard range an average agent will have of each resource, where it makes sense, e.g. it does not make sense for a resource like season, but it does for intelligence.

[1] It is possible to include all the sequences that has been evaluated, so there may be a few of the possible sequences available.

### 5.2 Activity Types, Activities, and Plans

The activity types along with all the activities in each activity type. For each activity, the resources that influence the motivation for the activity, which direction, and the strength of the influence along with the same for influence on the QoL of the activity. It is also a good idea to include a standard preference for both the activity types and the activities. Each activity should also be given a factor for each emotion, used for calculating the $EV$, by the Action Proposer. It may be good idea to let all $\overrightarrow{EF}$ for an agent sum to the same value, so that fine-tuning the behavior of agents can be done solely on the preferences.

Plans for acquiring resources should include the same as activities, with inconvenience replacing QoL. Furthermore the specification should include resources that influence the probability of the plan being successful, how they influence the probability, how resources are affected by attempting the plan, and how resources are affected by the plan being carried out successfully.

### 5.3 Agent Types and Agents

For each type of Agent the preferences of all of the above, where the agent type differs from an average agent, needs to be specified. If a preference is unspecified the standard preference will be used. The agent type can also include a standard range for the resources where an agent of that type will differ from the average agent. Agent types may also have $\overrightarrow{EF}$s that differ from the average agent, e.g. the bully type mentioned. It still makes sense to let $\overrightarrow{EF}$s sum to the same value, e.g. the bully type agent's inclination to beating people up will drop more than the average person's, when joy rises.

Each agent needs a type and a specification of the preferences where it differs from a standard agent of that type, note that preferences for activity/plan types are fixed inside an agent type. If some resource needs to be fixed at a specific value, or inside a range that differs from the average agent of that type, it must be specified. It is usually a good idea to make sure that the resources given to an agent fall within what is possible for that agent type to have. It is also possible to overwrite $\overrightarrow{EF}$s in the agent specification.

## Acknowledgments

The authors would in particular like to thank Nikolaj Hyldig for valuable discussions on his ideas on the EMS which we have given a normative angle. The authors would like to thank Hugin for providing a tool for specification of OOBNs. We would also like to thank the anonymous reviewers for valuable comments.

## References

A. Turing, 1950, *Computing machinery and intelligence*, Mind 59:433-460.

Blizzard Entertainment, 2000, *Diablo II*.

Bioware, 2002, *Neverwinter Nights*.

N. Hyldig, 2005, *EMS-20051228-2*, Twilight Project work document.

A. Ortony, G Clore, and A. Collins, 1988, *The Cognitive Structure of Emotions*, Cambridge University Press.

C. Bartneck, 2002, *Integrating the OCC Model of Emotions in Embodied Characters*, In Proceedings of the Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges, Melbourne.

O. Bangsø, P.H. Wuillemin, 2000, *Top-down Construction and Repetitive Structures Representation in Bayesian Networks* In Proceedings of the Thirteenth International Florida Artificial Intelligence Society Conference, pages 282-286.

# Loopy Propagation: the Convergence Error in Markov Networks

Janneke H. Bolt
Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
janneke@cs.uu.nl

**Abstract**

Loopy propagation provides for approximate reasoning with Bayesian networks. In previous research, we distinguished between two different types of error in the probabilities yielded by the algorithm; the cycling error and the convergence error. Other researchers analysed an equivalent algorithm for pairwise Markov networks. For such networks with just a simple loop, a relationship between the exact and the approximate probabilities was established. In their research, there appeared to be no equivalent for the convergence error, however. In this paper, we indicate that the convergence error in a Bayesian network is converted to a cycling error in the equivalent Markov network. Furthermore, we show that the prior convergence error in Markov networks is characterised by the fact that the previously mentioned relationship between the exact and the approximate probabilities cannot be derived for the loop node in which this error occurs.

## 1 Introduction

A Bayesian network uniquely defines a joint probability distribution and as such provides for computing any probability of interest over its variables. Reasoning with a Bayesian network, more specifically, amounts to computing (posterior) probability distributions for the variables involved. For networks without any topological restrictions, this reasoning task is known to be NP-hard (Cooper 1990). For networks with specific restricted topologies, however, efficient algorithms are available, such as Pearl's propagation algorithm for singly connected networks. Also the task of computing approximate probabilities with guaranteed error bounds is NP-hard in general (Dagum and Luby 1993). Although their results are not guaranteed to lie within given error bounds, various approximation algorithms are available that yield good results on many real-life networks. One of these algorithms is the *loopy-propagation algorithm*. The basic idea of this algorithm is to apply Pearl's propagation algorithm to a Bayesian network regardless of its topological structure. While the algorithm results in exact probability distributions for a singly connected network, it yields approximate probabilities for the variables of a multiply connected network. Good approximation performance has been reported for this algorithm (Murphy et al 1999).

In (Bolt and van der Gaag 2004), we studied the performance of the loopy-propagation from a theoretical point of view and argued that two types of error may arise in the approximate probabilities yielded by the algorithm: the *cycling error* and the *convergence error*. A cycling error arises when messages are being passed on within a loop repetitively and old information is mistaken for new by the variables involved. A convergence error arises when messages that originate from dependent variables are combined as if they were independent.

Many other researchers have addressed the performance of the loopy-propagation algorithm. Weiss and his co-workers, more specifically, investigated its performance by studying the application of an equivalent algorithm on pairwise Markov networks (Weiss 2000, and Weiss and Freeman 2001). Their use of Markov networks for this purpose was motivated by the relatively easier analysis of these networks and

justified by the observation that any Bayesian network can be converted into an equivalent pairwise Markov network. Weiss (2000) derived an analytical relationship between the exact and the computed probabilities for the loop nodes in a network including a single loop. In the analysis of loopy propagation in Markov networks, however, no distinction between different error types was made, and on first sight there is no equivalent for the convergence error. In this paper we investigate this difference in results; we do so by constructing the simplest situation in which a convergence error may occur, and analysing the equivalent Markov network. We find that the convergence error in the Bayesian Markov network is converted to a cycling error in the equivalent Markov network. Furthermore, we find that the prior convergence error in Markov networks is characterised by the fact that the relationship between the exact and the approximate probabilities, as established by Weiss, cannot be derived for the loop node in which this error occurs.

## 2 Bayesian Networks

A *Bayesian network* is a model of a joint probability distribution Pr over a set of stochastic variables $\mathbf{V}$, consisting of a directed acyclic graph and a set of conditional probability distributions[1]. Each variable $A$ is represented by a node $A$ in the network's digraph[2]. (Conditional) independency between the variables is captured by the digraph's set of arcs according to the d-separation criterion (Pearl 1988). The strength of the probabilistic relationships between the variables is captured by the conditional probability distributions $\Pr(A \mid \mathbf{p}(\mathbf{A}))$, where $\mathbf{p}(\mathbf{A})$ denotes the instantiations of the parents of $A$. The joint probability distribution is presented by

$$\Pr(\mathbf{V}) = \prod_{A \in \mathbf{V}} \Pr(A \mid \mathbf{p}(\mathbf{A}))$$

For the scope of this paper we assume all variables of a Bayesian network to be binary. We will often write $a$ for $A = a_1$ and $\bar{a}$ for $A = a_2$. Fig. 1 depicts a small binary Bayesian network.

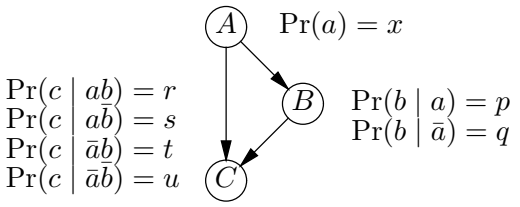



Figure 1: An example Bayesian network.

A multiply connected network includes one or more loops. We say that a loop is simple if none of its nodes are shared by another loop. A node that has two or more incoming arcs on a loop will be called a *convergence node* of this loop. Node $C$ is the only convergence node in the network from Fig. 1. Pearl's propagation algorithm (Pearl 1988) was designed for exact inference with singly connected Bayesian networks. The term *loopy propagation* used throughout the literature, refers to the application of this algorithm to networks with loops.

## 3 The Convergence Error in Bayesian Networks

When applied to a singly connected Bayesian network, Pearl's propagation algorithm results in exact probabilities. When applied to a multiply connected network, however, the computed probabilities may include errors. In previous work we distinguished between two different types of error (Bolt and van der Gaag 2004).

The first type of error arises when messages are being passed on in a loop repetitively and old information is mistaken for new by the variables involved. The error that thus arises will be termed a *cycling error*. A cycling error can only occur if for each convergence node of a loop either the node itself or one of its descendents is observed. The second type of error originates from the combination of causal messages by the

[1] Variables are denoted by upper-case letters ($A$), and their values by indexed lower-case letters ($a_i$); sets of variables by bold-face upper-case letters ($\mathbf{A}$) and their instantiations by bold-face lower-case letters ($\mathbf{a}$). The upper-case letter is also used to indicate the whole range of values of a variable or a set of variables.

[2] The terms node and variable will be used interchangeably.

convergence node of a loop. A convergence node combines the messages from its parents as if the parents are independent. They may be dependent, however, and by assuming independence, a *convergence error* may be introduced. A convergence error may already emerge in a network in its prior state. In the sequel we will denote the probabilities that result upon loopy propagation with $\widetilde{\Pr}$ to distinguish them from the exact probabilities which are denoted by Pr.

Moreover, we studied the prior convergence error. Below, we apply our analysis to the example network from Figure 1. For the network in its prior state, the loopy-propagation algorithm establishes

$$\widetilde{\Pr}(c) = \sum_{A,B} \Pr(c \mid AB) \cdot \Pr(A) \cdot \Pr(B)$$

as probability for node $C$. Nodes $A$ and $B$, however, may be dependent and the exact probability $\Pr(c)$ equals

$$\Pr(c) = \sum_{A,B} \Pr(c \mid AB) \cdot \Pr(B \mid A) \cdot \Pr(A)$$

The difference between the exact and approximate probabilities is

$$\Pr(c) - \widetilde{\Pr}(c) = x \cdot y \cdot z$$

where

$$\begin{aligned} x &= \Pr(c \mid ab) - \Pr(c \mid a\bar{b}) - \Pr(c \mid \bar{a}b) + \Pr(c \mid \bar{a}\bar{b}) \\ y &= \Pr(b \mid a) - \Pr(b \mid \bar{a}) \\ z &= \Pr(a) - \Pr(a)^2 \end{aligned}$$

The factors that govern the size of the prior convergence error in the network from Figure 1, are illustrated in Figure 2; for the construction of this figure we used the following probabilities: $r = 1$, $s = 0$, $t = 0$, $u = 1$, $p = 0.4$, $q = 0.1$ and $x = 0.5$. The line segment captures the exact probability $\Pr(c)$ as a function of $\Pr(a)$; note that each specific $\Pr(a)$ corresponds with a specific $\Pr(b)$. The surface captures $\widetilde{\Pr}(c)$ as a function of $\Pr(a)$ and $\Pr(b)$. The convergence error equals the distance between the point on the line segment that matches the probability $\Pr(a)$ from the network and its orthogonal projection on the surface. For $\Pr(a) = 0.5$, more specifically, the difference between $\Pr(c)$ and $\widetilde{\Pr}(c)$ is indicated by the vertical dotted line segment and equals $0.65 - 0.5 = 0.15$. Informally speaking:

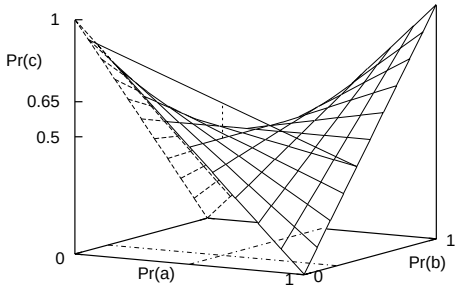



Figure 2: The probability of $c$ as a function of $\Pr(a)$ and $\Pr(b)$, assuming independence of the parents $A$ and $B$ of $C$ (surface), and as a function of $\Pr(a)$ (line segment).

- the more curved the surface is, the larger the distance between a point on the line segment and its projection on the surface can be; the curvature of the surface is reflected by the factor $x$;
- the distance between a point on the segment and its projection on the surface depends on the orientation of the line segment; the orientation of the line segment is reflected by the factor $y$;
- the distance between a point on the line segment and its projection on the surface depends its position on the line segment; this position is reflected by the factor $z$.

We recall that the convergence error originates from combining messages from dependent nodes as if they were independent. The factors $y$ and $z$ now in essence capture the degree of dependence between the nodes $A$ and $B$; the factor $x$ indicates to which extent this dependence can affect the computed probabilities.

## 4 Markov Networks

Like a Bayesian network, a Markov network uniquely defines a joint probability distribution over a set of statistical variables $\mathbf{V}$. The variables are represented by the nodes of an undirected graph and (conditional) independency between the variables is captured by the graph's set of edges; a variable is (conditionally) independent of every other variable given its Markov blanket. The strength of the probabilistic relationships is captured by clique potentials. Cliques $C$ are subsets of nodes that are completely connected; $\cup C = \mathbf{V}$. For each clique, a potential function $\psi_C(\mathbf{A}_C)$ is given that assigns a non-negative real number to each configuration of the nodes $\mathbf{A}$ of $C$. The joint probability is presented by:

$$\Pr(\mathbf{V}) = 1/Z \cdot \prod_C \psi_C(\mathbf{A}_C)$$

where $Z = \sum_{\mathbf{V}} \prod_C \psi_C(\mathbf{A}_C)$ is a normalising factor, ensuring that $\sum_{\mathbf{V}} \Pr(\mathbf{V}) = 1$. A pairwise Markov network is a Markov network with cliques of maximal two nodes.

For pairwise Markov networks, an algorithm can be specified that is functionally equivalent to Pearl's propagation algorithm (Weiss 2000). In this algorithm, in each time step, every node sends a probability vector to each of its neighbours. The probability distribution of a node is obtained by combining the steady state values of the messages from its neighbours.

In a pairwise Markov network, the transition matrices $M^{AB}$ and $M^{BA}$ can be associated with any edge between nodes $A$ and $B$.

$$M^{AB}_{ji} = \psi(A = a_i, B = b_j)$$

Note that matrix $M^{BA}$ equals $M^{AB^T}$.

**Example 1** Suppose we have a Markov network with two binary nodes $A$ and $B$, and suppose that for this network the potentials $\psi(ab) = p$, $\psi(a\bar{b}) = q$, $\psi(\bar{a}b) = r$ and $\psi(\bar{a}\bar{b}) = s$ are specified, as in Fig. 3 . We then associate the transition matrix $M^{AB} = \begin{bmatrix} p & r \\ q & s \end{bmatrix}$ with the link from $A$ to $B$ and its transpose $M^{BA} = \begin{bmatrix} p & q \\ r & s \end{bmatrix}$ with the link from $B$ to $A$. □

$\psi(ab) = p$
$\psi(a\bar{b}) = q$
$\psi(\bar{a}b) = r$
$\psi(\bar{a}\bar{b}) = s$

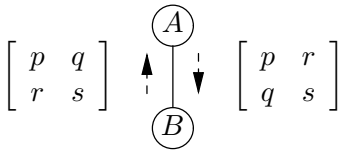



Figure 3: An example pairwise Markov network and its transition matrices.

The propagation algorithm now is defined as follows. The message from $A$ to $B$ equals $M^{AB} \cdot v$ after normalisation, where $v$ is the vector that results from the component wise multiplication of all message vectors sent to $A$ except for the message vector sent by $B$. The procedure is initialised with all message vectors set to (1,1,...,1). Observed nodes do not receive messages and they always transmit a vector with 1 for the observed value and zero for all other values. The probability distribution for a node, is obtained by combining all incoming messages, again by component wise multiplication and normalisation.

## 5 Converting a Bayesian Network into a Pairwise Markov Network

In this section, the conversion of a Bayesian network into an equivalent pairwise Markov network is described (Weiss 2000). In the conversion of the Bayesian network into a Markov network, for any node with multiple parents, an auxiliary node is constructed into which the common parents are clustered. This auxiliary node is connected to the child and its parents and the original arcs between child and parents are removed. Furthermore, all arc directions in the network are dropped. The clusters are all pairs of connected nodes. For a cluster with an auxiliary node and a former parent node, the potential is set to 1 if the nodes have a similar value for the former parent node and to 0 otherwise. For the other clusters, the potentials are equal to the conditional probabilities of the former child given the former parent. Furthermore, the prior probability of a former root node is incorporated by multiplication into one of the potentials of the clusters in which it takes part.

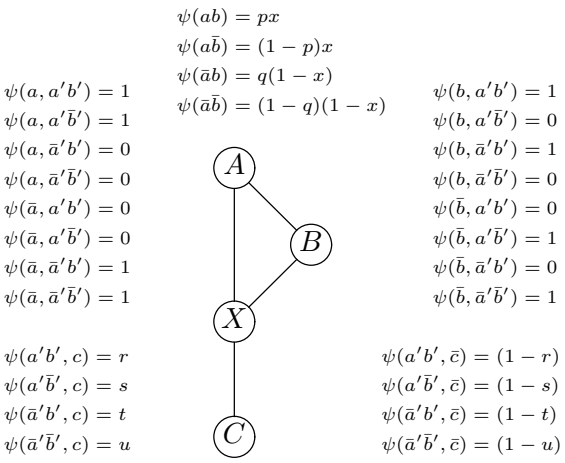



Figure 4: A pairwise Markov network that represents the same joint probability distribution as the Bayesian network from Figure 1.

**Example 2** The Bayesian network from Figure 1 can be converted into the pairwise Markov network from Figure 4 with clusters $AB$, $AX$, $BX$ and $XC$. Node $X$ is composed of $A'$ and $B'$ and has the values $a'b'$, $a'\bar{b}'$, $\bar{a}'b'$ and $\bar{a}'\bar{b}'$. Given that the prior probability of root node $A$ is incorporated in the potential of cluster $AB$, the network has the following potentials: $\psi(AB) = \Pr(B \mid A) \cdot \Pr(A)$; $\psi(XC) = \Pr(C \mid AB)$: $\psi(AX) = 1$ if $A' = A$ and 0 otherwise and; $\psi(BX) = 1$ if $B' = B$ and 0 otherwise.

## 6 The Analysis of Loopy Propagation in Markov Networks

Weiss (2000) analysed the performance of the loopy-propagation algorithm for Markov networks with a single loop and related the approximate probabilities found for the nodes in the loop to their exact probabilities. He noted that in the application of the algorithm messages will cycle in the loop and errors will emerge as a result of the double counting of information. The main idea of his analysis is that for a node in the loop, two reflexive matrices can be derived; one for the messages cycling clockwise and one for the messages cycling counterclockwise. The probability distribution computed by the loopy-propagation algorithm for the loop node in the steady state, now can be inferred from the principal eigenvectors of the reflexive matrices plus the other incoming vectors. Subsequently, he showed that the reflexive matrices also include the exact probability distribution and used those two observations to derive an analytical relationship between the approximated and the exact probabilities.

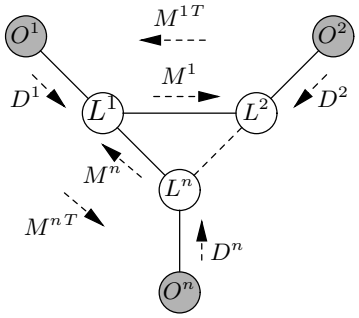



Figure 5: An example Markov network with just one loop.

More in detail, Weiss considered a Markov network with a single loop with $n$ nodes $L^1 ... L^n$ and with connected to each node in the loop, an observed node $O^1 ... O^n$ as shown in Figure 5. During propagation, a node $O^i$ will constantly send the same message into the loop. This vector is one of the columns of the transition matrix $M^{O^i L^i}$ In order to enable the incorporation of this message into the reflexive matrices, this vector is transformed into a diagonal matrix $D^i$, with the vector elements on the diagonal. For example, suppose that $M^{O^i L^i} = \begin{bmatrix} p & r \\ q & s \end{bmatrix}$ and suppose that the observation $O^i = o_1^i$ is made, then $D^i = \begin{bmatrix} p & 0 \\ 0 & q \end{bmatrix}$. Furthermore, $M^1$ is the transition matrix for the message from $L^1$ to $L^2$ and $M^{1^T}$ the transition matrix for the message from $L^2$ to $L^1$ etc. The reflexive matrix $C$ for the transition of a counterclockwise message from node $L^1$ back to itself is defined as $M^{1^T} D^2 ... M^{n-1^T} D^n M^{n^T} D^1$. The message that $L^2$ sends to $L^1$ in the steady state now is in the direction of the principal eigenvector of $C$. The reflexive matrix $C^2$ for the transition of a clockwise message from node $L^1$ back to itself is defined as $M^n D^n M^{n-1} D^{n-1} ... M^1 D^1$. The message that node $L^n$ sends to $L^1$ in the steady state is in the direction of the principal eigenvector of $C^2$. Component wise multiplication of

the two principal eigenvectors and the message from $O^1$ to $L^1$, and normalisation of the resulting vector, yields a vector of which the components equal the approximated values for $L^1$ in the steady state. Furthermore, Weiss proved that the elements on the diagonals of the reflexive matrices equal the correct probabilities of the relevant value of $L_1$ and the evidence, for example, $C_{1,1}$ equals $\Pr(l_1^1, \mathbf{o})$. Subsequently, he related the exact probabilities for a node $A$ in the loop to its approximate probabilities by

$$\Pr(a_i) = \frac{\lambda_1 \widetilde{\Pr}(a_i) + \sum_{j=2} P_{ij}\lambda_j P_{ji}^{-1}}{\sum_j \lambda_j} \qquad (1)$$

in which $P$ is a matrix that is composed of the eigenvectors of $C$, with the principal eigenvector in the first column, and $\lambda_1 ... \lambda_j$ are the eigenvalues of the reflexive matrices, with $\lambda_1$ the maximum eigenvalue. We note that from this formula it follows that correct probabilities will be found if $\lambda_1$ equals 1 and all other eigenvalues equal 0.

In the above analysis, all nodes $O^i$ are considered to be observed. Note that given unobserved nodes outside the loop, the analysis is essentially the same. In that case a transition matrix $M^{O^i L^i} = \begin{bmatrix} p & r \\ q & s \end{bmatrix}$ will result in the diagonal matrix $D^i = \begin{bmatrix} p+r & 0 \\ 0 & q+s \end{bmatrix}$.

## 7 The Convergence Error in Markov Networks

As discussed in Section 3 in Bayesian networks, a distinction could be made between the cycling error and the convergence error. In the previous section it appeared that for Markov network such a distinction does not exist. All errors result from the cycling of information and, on first sight, there is no equivalent for the convergence error. However, any Bayesian network can be converted into an equivalent pairwise Markov network on which an algorithm equivalent to the loopy-propagation algorithm can be used. In this section, we investigate this apparent incompatibility of results and indicate how the convergence error yet is embedded in the analysis of loopy propagation in Markov networks. We do so by constructing the simplest situation in which a convergence error may occur, that is, the Bayesian network from Figure 1 in its prior state, and analysing this situation in the equivalent Markov network. The focus thereby is on the node that replaces the convergence node in the loop. We then argue that the results have a more general validity.

Consider the Bayesian network from Figure 1. In its prior state, there is no cycling of information, and exact probabilities will be found for nodes $A$ and $B$. In node $C$, however, a convergence error may emerge. The network can be converted into the pairwise Markov network from Figure 4. For this network we find the transition matrices $M^{XA} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$, $M^{XB} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$, $M^{XC} = \begin{bmatrix} r & s & t & u \\ 1-r & 1-s & 1-t & 1-u \end{bmatrix}$, $M^{AB} = \begin{bmatrix} px & q(1-x) \\ (1-p)x & (1-p)(1-x) \end{bmatrix}$ and their transposes. In the prior state of the network, $C$ will send the message $M^{CX} \cdot (1,1) = (1,1,1,1)$ to $X$. In order to enable the incorporation of this message into the reflexive loop matrices it is transformed into $D^{CX}$ which, in this case, is the 4x4-identity matrix.

We first evaluate the performance of the loopy propagation algorithm for the regular loop node $A$. This node has the following reflexive matrices for its clockwise and counterclockwise messages respectively:

$$M^{\circlearrowleft A} = M^{XA} \cdot D^{CX} \cdot M^{BX} \cdot M^{AB} = \begin{bmatrix} x & 1-x \\ x & 1-x \end{bmatrix}$$

with eigenvalues 1 and 0 and principal eigenvector (1,1) and

$$M^{\circlearrowleft A} = M^{BA} \cdot M^{XB} \cdot D^{CX} \cdot M^{AX} = \begin{bmatrix} x & x \\ 1-x & 1-x \end{bmatrix}$$

with eigenvalues 1 and 0 and principal eigenvector $(x, 1-x)$. Note that the correct probabilities for node $A$ indeed are found on the diagonal of the reflexive matrices. Furthermore, $\lambda_1 = 1$ and $\lambda_2 = 0$ and therefore correct approximations are

expected. We indeed find that the approximations $(1 \cdot x, 1 \cdot (1-x))$ equal the exact probabilities. Note also that, as expected, the messages from node $A$ back to itself do not change any more after the first cycle. As in the Bayesian network, for node $A$ no cycling of information occurs in the Markov network. For node $B$ a similar evaluation can be made.

We now turn to the convergence node $C$. In the Bayesian network in its prior state a convergence error may emerge in this node. In the conversion of the Bayesian network into the Markov network, the convergence node $C$ is placed outside the loop and the auxiliary node $X$ is added. For $X$, the following reflexive matrices are computed for the clockwise and counterclockwise messages from node $X$ back to itself respectively:

$$M^{\circlearrowright X} = M^{BX} \cdot M^{AB} \cdot M^{XA} \cdot D^{CX} =$$
$$\begin{bmatrix} px & px & q(1-x) & q(1-x) \\ (1-p)x & (1-p)x & (1-q)(1-x) & (1-q)(1-x) \\ px & px & q(1-x) & q(1-x) \\ (1-p)x & (1-p)x & (1-q)(1-x) & (1-q)(1-x) \end{bmatrix}$$

with eigenvalues 1, 0, 0, 0; principal eigenvector $((px+q(1-x))/((1-p)x+(1-q)(1-x)), 1, (px+q(1-x))/((1-p)x+(1-q)(1-x)), 1)$ and other eigenvectors $(0, 0, -1, 1)$, $(-1, 1, 0, 0)$ and $(0, 0, 0, 0)$.

$$M^{\circlearrowleft X} = M^{AX} \cdot M^{BA} \cdot M^{XB} \cdot M^{CX} =$$
$$\begin{bmatrix} px & (1-p)x & px & (1-p)x \\ px & (1-p)x & px & (1-p)x \\ q(1-x) & (1-q)(1-x) & q(1-x) & (1-q)(1-x) \\ q(1-x) & (1-q)(1-x) & q(1-x) & (1-q)(1-x) \end{bmatrix}$$

with eigenvalues 1, 0, 0, 0; principal eigenvector $(x/(1-x), x/(1-x), 1, 1)$ and other eigenvectors $(0, -1, 0, 1)$, $(-1, 0, 1, 0)$ and $(0, 0, 0, 0)$.

On the diagonal of the reflexive matrices of $X$ we find the probabilities $\Pr(AB)$. As the correct probabilities for a loop node are found on the diagonal of its reflexive matrices, these probabilities can be considered to be the exact probabilities for node $X$. The normalised vector of the component wise multiplication of the principal eigenvectors of the two reflexive matrices of $X$ equals the vector with the normalised probabilities $\Pr(A) \cdot \Pr(B)$. Likewise, these probabilities can be considered to be the approximate probabilities for node $X$.

A first observation is that $\lambda_1$ equals 1 and the other eigenvalues equal 0, but the exact and the approximate probabilities of node $X$ may differ. This is not consistent with Equation 1. The explanation is that for node $X$, the matrix $P$, is singular and therefore, the matrix $P^{-1}$, which is needed in the derivation of the relationship between the exact and approximate probabilities, does not exist. Equation 1, thus isn't valid for the auxiliary node $X$. We note furthermore that the messages from node $X$ back to itself may still change after the first cycle. We therefore find that, although in the Bayesian network there is no cycling of information, in the Markov network, for node $X$ information may cycle, resulting in errors computed for its probabilities.

The probabilities computed by the loopy-propagation algorithm for node $C$ equal the normalised product $M^{XC} \cdot v$, where $v$ is the vector with the approximate probabilities found at node $X$. It can easily be seen that these approximate probabilities equal the approximate probabilities found in the equivalent Bayesian network. Furthermore we observe that if node $X$ would send its exact probabilities, that is, $\Pr(AB)$, exact probabilities for node $C$ would be computed. In the Markov network we thus may consider the convergence error to be founded in the cycling of information for the auxiliary node $X$.

In Section 3, a formula for the size of the prior convergence error in the network from figure 1 is given. We there argued that this size is determined by the factors $y$ and $z$ that capture the degree of dependency between the parents of the convergence node and the factor $x$, that indicates to which extent the dependence between nodes $A$ and $B$ can affect the computed probabilities. In this formula, $x$ is composed of the conditional probabilities of node $C$. In the analysis in the Markov network we have a similar finding. The effect of the degree of dependence between $A$ and $B$ is reflected in the difference between the exact and the approximate probabilities found for node $X$. The effect of the conditional probabilities at node $C$ emerges in the transition of the message vector from $X$ to $C$.

We just considered the small example network from Figure 1. Note, however, that for any prior binary Bayesian networks with just simple loops, the situation for any loop can be 'summarised' to the situation in Figure 1 by marginalisation over the relevant variables. The results with respect to the manifestation of the convergence error by the cycling of information and the invalidity of Equation 1 for the auxiliary node, found for the network from Figure 1, therefore, apply to any prior binary Bayesian networks with just simple loops.[3][4]

## 8 Discussion

Loopy propagation refers to the application of Pearl's propagation algorithm for exact reasoning with singly connected Bayesian networks to networks with loops. In previous research we identified two different types of error that may arise in the probabilities computed by the algorithm. Cycling errors result from the cycling of information and arise in loop nodes as soon as for each convergence node of the loop, either the node itself, or one of its descendents is observed. Convergence errors result from combining information from dependent nodes as if they were independent and may arise at convergence nodes. This second error type is found both in a network's prior and posterior state. Loopy propagation has also been studied by the analysis of the performance of an equivalent algorithm in pairwise Markov networks with just a simple loop. According to this analysis all errors result from the cyling of information and on first sight there is no equivalent for the convergence error. We investigated how the convergence error yet is embedded in the analysis of loopy propagation in Markov networks. We did so by constructing the simplest situation in which a convergence error may occur, and analysing this situation in the equivalent Markov network. We found that the convergence error in the Bayesian network is converted to a cycling error in the equivalent Markov network. Furthermore, we found that the prior convergence error is characterised by the fact that the relationship between the exact probabilities and the approximate probabilities yielded by loopy propagation, as established by Weiss, can not be derived for the loop node in which this error occurs. We then argued that these results are valid for binary Bayesian network with just simple loops in general.

[3] Given a loop with multiple convergence nodes, in the prior state of the network, the parents of the convergence nodes are independent and effectively no loop is present.

[4] Two loops in sequence may result in incorrect probabilities entering the second loop. The reflexive matrices, however, will have a similar structure as the reflexive matrices derived in this section.

**Acknowledgements**

This research was partly supported by the Netherlands Organisation for Scientific Research (NWO).

## References

J.H. Bolt, L.C. van der Gaag. 2004. The convergence error in loopy propagation. Paper presented at *the International Conference on Advances in Intelligent Systems: Theory and Applications*.

G.F. Cooper. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405.

P. Dagum, M. Luby. 1993. Approximate inference in Bayesian networks is NP hard. *Artificial Intelligence*, 60:141–153.

K. Murphy, Y. Weiss, M. Jordan. 1999. Loopy belief propagation for approximate inference: an empirical study. In *15th Conference on Uncertainty in Artificial Intelligence*, pages 467–475.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Palo Alto.

Y. Weiss. 2000. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41.

Y. Weiss, W.T. Freeman. 2001. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13:2173–2200.

# Preprocessing the MAP problem

Janneke H. Bolt and Linda C. van der Gaag
Department of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{janneke,linda}@cs.uu.nl

**Abstract**

The MAP problem for Bayesian networks is the problem of finding for a set of variables an instantiation of highest posterior probability given the available evidence. The problem is known to be computationally infeasible in general. In this paper, we present a method for preprocessing the MAP problem with the aim of reducing the runtime requirements for its solution. Our method exploits the concepts of Markov and MAP blanket for deriving partial information about a solution to the problem. We investigate the practicability of our preprocessing method in combination with an exact algorithm for solving the MAP problem for some real Bayesian networks.

## 1 Introduction

Upon reasoning with a Bayesian network, often a best explanation is sought for a given set of observations. Given the available evidence, such an explanation is an instantiation of highest probability for some subset of the network's variables. The problem of finding such an instantiation has an unfavourable computational complexity. If the subset of variables for which a most likely instantiation is to be found includes just a single variable, then the problem, which is known as the Pr problem, is NP-hard in general. A similar observation holds for the MPE problem in which an instantiation of highest probability is sought for all unobserved variables. These two problems can be solved in polynomial time, however, for networks of bounded treewidth. If the subset of interest is a non-singleton proper subset of the set of unobserved variables, on the other hand, the problem, which is then known as the MAP problem, remains NP-hard even for networks for which the other two problems can be feasibly solved (Park and Darwiche, 2002).

By performing inference in a Bayesian network under study and establishing the most likely value for each variable of interest separately, an estimate for a solution to the MAP problem may be obtained. There is no guarantee in general, however, that the values in the resulting joint instantiation indeed correspond to the values of the variables in a solution to the MAP problem. In this paper, we now show that, for some of the variables of interest, the computation of marginal posterior probabilities may in fact provide exact information about their value in a solution to the MAP problem. We show more specifically that, by building upon the concept of Markov blanket, some of the variables may be fixed to a particular value; for some of the other variables of interest, moreover, values may be excluded from further consideration. We further introduce the concept of MAP blanket that serves to provide similar information.

Deriving partial information about a solution to the MAP problem by building upon the concepts of Markov and MAP blanket, can be exploited as a preprocessing step before the problem is actually solved with any available algorithm. The derived information in essence serves to reduce the search space for the problem and thereby reduces the algorithm's runtime requirements. We performed an initial study of the practicability of our preprocessing method by solving MAP problems for real networks using an exact branch-and-bound al-

gorithm, and found that preprocessing can be profitable.

The paper is organised as follows. In Section 2, we provide some preliminaries on the MAP problem. In Section 3, we present two propositions that constitute the basis of our preprocessing method. In Section 4, we provide some preliminary results about the practicability of our preprocessing method. The paper is ended in Section 5 with our concluding observations.

## 2 The MAP problem

Before reviewing the MAP problem, we introduce our notational conventions. A Bayesian network is a model of a joint probability distribution Pr over a set of stochastic variables, consisting of a directed acyclic graph and a set of conditional probability distributions. We denote variables by upper-case letters ($A$) and their values by (indexed) lower-case letters ($a_i$); sets of variables are indicated by bold-face upper-case letters ($\mathbf{A}$) and their instantiations by bold-face lower-case letters ($\mathbf{a}$). Each variable is represented by a node in the digraph; (conditional) independence between the variables is encoded by the digraph's set of arcs according to the d-separation criterion (Pearl, 1988). The Markov blanket $\mathbf{B}$ of a variable $A$ consists of its neighbours in the digraph plus the parents of its children. Given its Markov blanket, the variable is independent of all other variables in the network. The strengths of the probabilistic relationships between the variables are captured by conditional probability tables that encode for each variable $A$ the conditional distributions $\Pr(A \mid \mathbf{p}(\mathbf{A}))$ given its parents $\mathbf{p}(\mathbf{A})$.

Upon reasoning with a Bayesian network, often a best explanation is sought for a given set of observations. Given evidence $\mathbf{o}$ for a subset of variables $\mathbf{O}$, such an explanation is an instantiation of highest probability for some subset $\mathbf{M}$ of the network's variables. The set $\mathbf{M}$ is called the *MAP set* for the problem; its elements are called the *MAP variables*. An instantiation $\mathbf{m}$ of highest probability to the set $\mathbf{M}$ is termed a *MAP solution*; the value that is assigned to a MAP variable in a solution $\mathbf{m}$ is called its *MAP value*. Dependent upon the size of the MAP set, we distinguish between three different types of problem. If the MAP set includes just a single variable, the problem of finding the best explanation for a set of observations reduces to establishing the most likely value for this variable from its marginal posterior probability distribution. This problem is called the Pr *problem* as it essentially amounts to performing standard inference (Park and Darwiche, 2001). In the second type of problem, the MAP set includes *all* non-observed variables. This problem is known as the most probable explanation or *MPE problem*. In this paper, we are interested in the third type of problem, called the *MAP problem*, in which the MAP set is a non-singleton proper subset of the set of non-observed variables of the network under study. This problem amounts to finding an instantiation of highest probability for a designated set of variables of interest.

We would like to note that the MAP problem is more complex in essence than the other two problems. The Pr problem and the MPE problem both are NP-hard in general and are solvable in polynomial time for Bayesian networks of bounded treewidth. The MAP problem is $\mathrm{NP}^{\mathrm{PP}}$-hard in general and remains NP-hard for these restricted networks (Park, 2002).

## 3 Fixing MAP values

By performing inference in a Bayesian network and solving the Pr problem for each MAP variable separately, an estimate for a MAP solution may be obtained. There is no guarantee in general, however, that the value with highest marginal probability for a variable corresponds with its value in a MAP solution. We now show that, for some variables, the computation of marginal probabilities may in fact provide exact information about their MAP values.

The first property that we will exploit in the sequel, builds upon the concept of Markov blanket. We consider a MAP variable $H$ and its associated Markov blanket. If a specific value $h_i$ of $H$ has highest probability in the marginal distribution over $H$ for all possible instantia-

tions of the blanket, then $h_i$ will be the value of $H$ in a MAP solution for any MAP problem including $H$. Alternatively, if some value $h_j$ never has highest marginal probability, then this value cannot be included in any solution.

**Proposition 1.** *Let $H$ be a MAP variable in a Bayesian network and let $\mathbf{B}$ be its Markov blanket. Let $h_i$ be a specific value of $H$.*

1. *If $\Pr(h_i \mid \mathbf{b}) \geq \Pr(h_k \mid \mathbf{b})$ for all values $h_k$ of $H$ and all instantiations $\mathbf{b}$ of $\mathbf{B}$, then $h_i$ is the value of $H$ in a MAP solution for any MAP problem that includes $H$.*

2. *If there exist values $h_k$ of $H$ with $\Pr(h_i \mid \mathbf{b}) < \Pr(h_k \mid \mathbf{b})$ for all instantiations $\mathbf{b}$ of $\mathbf{B}$, then $h_i$ is not the MAP value of $H$ in any solution to a MAP problem that includes $H$.*

**Proof.** We prove the first property stated in the proposition; the proof of the second property builds upon similar arguments.

We consider an arbitrary MAP problem with the MAP set $\{H\} \cup \mathbf{M}$ and the evidence $\mathbf{o}$ for the observed variables $\mathbf{O}$. Finding a solution to the problem amounts to finding an instantiation to the MAP set that maximises the posterior probability $\Pr(H, \mathbf{M} \mid \mathbf{o})$. We have that

$$\Pr(h_i, \mathbf{M} \mid \mathbf{o}) =$$
$$= \textstyle\sum_{\mathbf{b}} \Pr(h_i \mid \mathbf{b}, \mathbf{o}) \cdot \Pr(\mathbf{M} \mid \mathbf{b}, \mathbf{o}) \cdot \Pr(\mathbf{b} \mid \mathbf{o})$$

For the posterior probability $\Pr(h_k, \mathbf{M} \mid \mathbf{o})$ an analogous expression is found. Now suppose that for the value $h_i$ of $H$ we have that $\Pr(h_i \mid \mathbf{b}) \geq \Pr(h_k \mid \mathbf{b})$ for all values $h_k$ of $H$ and all instantiations $\mathbf{b}$ of $\mathbf{B}$. Since $\mathbf{B}$ is the Markov blanket of $H$, we have that $\Pr(h_i \mid \mathbf{b}) \geq \Pr(h_k \mid \mathbf{b})$ implies $\Pr(h_i \mid \mathbf{b}, \mathbf{o}) \geq \Pr(h_k \mid \mathbf{b}, \mathbf{o})$ for all values $h_k$ of $H$ and all instantiations $\mathbf{b}$ of $\mathbf{B}$. We conclude that $\Pr(h_i, \mathbf{M} \mid \mathbf{o}) \geq \Pr(h_k, \mathbf{M} \mid \mathbf{o})$ for all $h_k$. The value $h_i$ of $H$ thus is included in a solution to the MAP problem under study. Since the above considerations are algebraically independent of the MAP variables $\mathbf{M}$ and of the evidence $\mathbf{o}$, this property holds for any MAP problem that includes the variable $H$. □

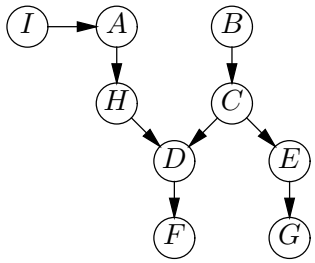



Figure 1: An example directed acyclic graph.

The above proposition provides for preprocessing a MAP problem. Prior to actually solving the problem, some of the MAP variables may be fixed to a particular value using the first property. With the second property, moreover, various values of the MAP variables may be excluded from further consideration. By building upon the proposition, therefore, the search space for the MAP problem is effectively reduced. We illustrate this with an example.

**Example 1.** We consider a Bayesian network with the graphical structure from Figure 1. Suppose that $H$ is a ternary MAP variable with the values $h_1$, $h_2$ and $h_3$. The Markov blanket $\mathbf{B}$ of $H$ consists of the three variables $A$, $C$ and $D$. Now, if for any instantiation $\mathbf{b}$ of these variables we have that $\Pr(h_1 \mid \mathbf{b}) \geq \Pr(h_2 \mid \mathbf{b})$ and $\Pr(h_1 \mid \mathbf{b}) \geq \Pr(h_3 \mid \mathbf{b})$, then $h_1$ occurs in a MAP solution for any MAP problem that includes $H$. By fixing the variable $H$ to the value $h_1$, the search space of any such problem is reduced by a factor 3. We consider, as an example, the MAP set $\{H, E, G, I\}$ of ternary variables. Without preprocessing, the search space includes $3^4 = 81$ possible instantiations. By fixing the variable $H$ to $h_1$, the search space of the problem reduces to $3^3 = 27$ instantiations. □

Establishing whether or not the properties from Proposition 1 can be used for a specific MAP variable, requires a number of computations that is exponential in the size of the variable's Markov blanket. The computations required, however, are highly local. A single restricted inward propagation for each instantiation of the Markov blanket to the variable of interest suffices. Since the proposition moreover holds for

any MAP problem that includes the variable, the computational burden involved is amortised over all future MAP computations.

The second proposition that we will exploit for preprocessing a MAP problem, builds upon the new concept of *MAP blanket*. We consider a problem with the MAP variables $\{H\} \cup \mathbf{M}$. A MAP blanket $\mathbf{K}$ of $H$ now is a minimal subset $\mathbf{K} \subseteq \mathbf{M}$ such that $\mathbf{K}$ d-separates $H$ from $\mathbf{M} \setminus \mathbf{K}$ given the available evidence. Now, if a specific value $h_i$ of $H$ has highest probability in the marginal distribution over $H$ given the evidence for all possible instantiations of $\mathbf{K}$, then $h_i$ will be the MAP value of $H$ in a solution to the MAP problem under study. Alternatively, if some value $h_j$ never has highest marginal probability, then this value cannot be a MAP value in any of the problem's solutions.

**Proposition 2.** *Let $\{H\} \cup \mathbf{M}$ be the MAP set of a given MAP problem for a Bayesian network and let $\mathbf{o}$ be the evidence that is available for the observed variables $\mathbf{O}$. Let $\mathbf{K}$ be the MAP blanket for the variable $H$ given $\mathbf{o}$, and let $h_i$ be a specific value of $H$.*

1. *If $\Pr(h_i \mid \mathbf{k}, \mathbf{o}) \geq \Pr(h_k \mid \mathbf{k}, \mathbf{o})$ for all values $h_k$ of $H$ and all instantiations $\mathbf{k}$ to $\mathbf{K}$, then $h_i$ is the MAP value of $H$ in a solution to the given MAP problem.*

2. *If there exist values $h_k$ of $H$ with $\Pr(h_i \mid \mathbf{k}, \mathbf{o}) < \Pr(h_k \mid \mathbf{k}, \mathbf{o})$ for all instantiations $\mathbf{k}$ to $\mathbf{K}$, then $h_i$ is not the MAP value of $H$ in any solution to the given MAP problem.*

The proof of the proposition is relatively straightforward, building upon similar arguments as the proof of Proposition 1.

The above proposition again provides for preprocessing a MAP problem. Prior to actually solving the problem, the values of some variables may be fixed and other values may be excluded for further consideration. The proposition therefore again serves to effectively reduce the search space for the problem under study. While the information derived from Proposition 1 holds for any MAP problem including $H$, however, Proposition 2 provides information for any problem in which $H$ has a subset of $\mathbf{K}$ for its MAP blanket and with matching evidence for the observed variables that are not d-separated from $H$ by $\mathbf{K}$. The information derived from Proposition 2, therefore, is more restricted in scope than that from Proposition 1.

We illustrate the application of Proposition 2 for our example network.

**Example 2.** We consider again the MAP set $\{H, E, G, I\}$ for the Bayesian network from Example 1. In the absence of any evidence, the MAP blanket of the variable $H$ includes just the variable $I$. Now, if for each value $i$ of $I$ we have that $\Pr(h_1 \mid i) \geq \Pr(h_2 \mid i)$ and $\Pr(h_1 \mid i) \geq \Pr(h_3 \mid i)$, then the value $h_1$ occurs in a solution to the given MAP problem. The search space for actually solving the problem thus again is reduced from 81 to 27. □

Establishing whether or not the properties from Proposition 2 can be used for a specific MAP variable, requires a number of computations that is exponential in the size of the variable's MAP blanket. The size of this blanket is strongly dependent of the network's connectivity and of the location of the various MAP variables and observed variables in the network. The MAP blanket can in fact be larger in size than the Markov blanket of the variable. The computations required, moreover, are less local than those required for Proposition 1 and can involve full inward propagations to the variable of interest. Since the proposition in addition applies to just a restricted class of MAP problems, the computational burden involved in its verification can be amortised over other MAP computations to a lesser extent than that involved in the verification of Proposition 1.

The general idea underlying the two propositions stated above is the same. The idea is to verify whether or not a particular value of $H$ can be fixed or excluded as a MAP value by investigating $H$'s marginal probability distributions given all possible instantiations of a collection of variables surrounding $H$. Proposition 1

uses for this purpose the Markov blanket of $H$. By building upon the Markov blanket, which in essence is independent of the MAP set and of the entered evidence, generally applicable statements about the values of $H$ are found. A disadvantage of building upon the Markov blanket, however, is that maximally different distributions for $H$ are examined, which decreases the chances of fixing or excluding values.

By taking a blanket-like collection of variables at a larger distance from the MAP variable $H$, the marginal distributions examined for $H$ are likely to be less divergent, which serves to increase the chances of fixing or excluding values of $H$ as MAP values. A major disadvantage of such a blanket, however, is that its size tends to grow with the distance from $H$, which will result in an infeasibly large number of instantiations to be studied. For any blanket-like collection, we observe that the MAP variables of the problem have to either be in the blanket or be d-separated from $H$ by the blanket. Proposition 2 builds upon this observation explicitly and considers only the instantiations of the MAP blanket of the variable. The proposition thereby reduces the computations involved in its application yet retains and even further exploits the advantage of examining less divergent marginal distributions over $H$. Note that the values that can be fixed or excluded based on the first proposition, will also be fixed or excluded based on the second proposition. It may nevertheless still be worthwhile to exploit the first proposition because, as stated before, with this proposition values can be fixed or excluded in general and more restricted and possibly less computations are required.

So far we have argued that application of the two propositions serves to reduce the search space for a MAP problem by fixing variables to particular values and by excluding other values from further consideration. We would like to mention that by fixing variables the graphical structure of the Bayesian network under study may fall apart into unconnected components, for which the MAP problem can be solved separately. We illustrate the basic idea with our running example.

**Example 3.** We consider again the MAP set $\{H, E, G, I\}$ for the Bayesian network from Figure 1. Now suppose that the variable $H$ can be fixed to a particular value. Then, by performing evidence absorption of this value, the graphical structure of the network falls apart into the two components $\{A, I, H\}$ and $\{B, C, D, E, F, G\}$, respectively. The MAP problem then decomposes into the problem with the MAP set $\{I\}$ for the first component and the problem with the MAP set $\{E, G\}$ for the second component; both these problems now include the value of $H$ as further evidence. The search space thus is further reduced from 27 to $3 + 9 = 12$ instantiations to be studied. □

## 4 Experiments

In the previous section, we have introduced a method for preprocessing the MAP problem for Bayesian networks. In this section, we perform a preliminary study of the practicability of our method by solving MAP problems for real networks using an exact algorithm. In Section 4.1 we describe the set-up of the experiments; we review the results in Section 4.2.

### 4.1 The Experimental Set-up

In our experiments, we study the effects of our preprocessing method on three real Bayesian networks. We first report the percentages of values that are fixed or excluded by exploiting Proposition 1. We then compare the numbers of fixed variables as well as the numbers of network propagations with and without preprocessing, upon solving various MAP problems with a state-of-the-art exact algorithm.

In our experiments, we use three real Bayesian networks with a relatively high connectivity; Table 1 reports the numbers of variables and values for these networks. The *Wilson's disease* network (WD) is a small network in medicine, developed for the diagnosis of Wilson's liver disease (Korver and Lucas, 1993). The *classical swine fever* network (CSF) is a network in veterinary science, currently under development, for the early detection of outbreaks of classical swine fever in pig herds

(Geenen and Van der Gaag, 2005). The extended *oesophageal cancer* network (OESO+) is a moderately-sized network in medicine, which has been developed for the prediction of response to treatment of oesophageal cancer (Aleman et al, 2000). For each network, we compute MAP solutions for randomly generated MAP sets with 25% and 50% of the network's variables, respectively; for each size, five sets are generated. We did not set any evidence.

For solving the various MAP problems in our experiments, we use a basic implementation of the exact branch-and-bound algorithm available from Park and Darwiche (2003). This algorithm solves the MAP problem exactly for most networks for which the Pr and MPE problems are feasible. The algorithm constructs a depth-first search tree by choosing values for subsequent MAP variables, cutting off branches using an upper bound. Since our preprocessing method reduces the search space by fixing variables and excluding values, it essentially serves to decrease the depth of the tree and to diminish its branching factor.

### 4.2 Experimental results

In the first experiment, we established for each network the number of values that could be fixed or excluded by applying Proposition 1, that is, by studying the marginal distributions per variable given its Markov blanket. For computational reasons, we decided not to investigate variables for which the associated blanket had more than 45 000 different instantiations; this number is arbitrarily chosen. In the WD, CSF and OESO+ networks, there were 0, 8 and 15 of such variables respectively.

The results of the first experiment are presented in Table 1. The table reports, for each network, the total number of variables, the total number of values, the number of variables for which a value could be fixed, and the number of values that could be fixed or excluded; note that if, for example, for a ternary variable a value can be fixed, then also two values can be excluded. We observe that 17.1% to 19.0% of the variables could be fixed to a particular value. The number of values that could be fixed or excluded ranges between 15.3% and 23.2%. We would like to stress that whenever a variable can be fixed to a particular value, this result is valid for any MAP problem that includes this variable. The computations involved, therefore, have to be performed only once.

Table 1: The numbers of fixed and excluded values.

| *network* | *#vars.* | *#vals.* | *#vars.f.* | *#vals.f.+e.* |
|---|---|---|---|---|
| WD | 21 | 56 | 4(19.0%) | 13(23.2%) |
| CSF | 41 | 98 | 7(17.1%) | 15(15.3%) |
| OESO+ | 67 | 175 | 12(17.9%) | 27(15.4%) |

In the second experiment, we compared for each network the numbers of variables that could be fixed by the two different preprocessing steps; for Proposition 2, we restricted the number of network propagations per MAP variable to four because of the limited applicability of the resulting information. We further established the numbers of network propagations of the exact branch-and-bound algorithm that were forestalled by the preprocessing.

The results of the second experiment are presented in Table 2. The table reports in the two leftmost columns, for each network, the sizes of the MAP sets used and the average number of network propagations performed without any preprocessing. In the subsequent two columns, it reports the average number of variables that could be fixed to a particular value by using Proposition 1 and the average number of network propagations performed by the branch-and-bound algorithm after this preprocessing step. In the fifth and sixth columns, the table reports the numbers obtained with Proposition 2; the sixth column in addition shows, between parenthesis, the average number of propagations that are required for the application of Proposition 2. In the final two columns of the table, results for the two preprocessing steps combined are reported: the final but one column again mentions the number of variables that could be fixed to a particular value by Proposition 1; it moreover mentions the average number of variables that could be fixed to a particular value by Proposition 2 after Proposition 1

had been used. The rightmost column reports the average number of network propagations required by the branch-and-bound algorithm. We would like to note that the additional computations required for Proposition 2 are restricted network propagations; although the worst-case complexity of these propagations is the same as that of the propagations performed by the branch-and-bound algorithm, their runtime requirements may be considerably less.

From Table 2, we observe that the number of network propagations performed by the branch-and-bound algorithm grows with the number of MAP variables, as expected. For the two smaller networks, we observe in fact that the number of propagations without preprocessing equals the number of MAP variables plus one. For the larger OESO+ network, the number of network propagations performed by the algorithm is much larger than the number of MAP variables. This finding is not unexpected since the MAP problem has a high computational complexity. For the smaller networks, we further observe that each variable that is fixed by one of the preprocessing steps translates directly into a reduction of the number of propagations by one. For the OESO+ network, we find a larger reduction in the number of network propagations per fixed variable. Our experimental results thus indicate that the number of propagations required by the branch-and-bound algorithm indeed is decreased by fixing variables to their MAP value.

With respect to using Proposition 1, we observe that in all networks under study a reasonably number of variables could be fixed to a MAP value. We did not take the number of local propagations required for this proposition into consideration because the computational burden involved is amortised over future MAP computations. With respect to using Proposition 2, we observe that for all networks and all MAP sets the additional computations involved outweigh the number of network computations that are forestalled for the algorithm. This observation applies to using just Proposition 2 as well as to using the proposition after Proposition 1 has been applied. We also observe that fewer variables are fixed in the step based on Proposition 2 than in the step based on Proposition 1. This can be attributed to the limited number of propagations used in the step based on Proposition 2. We conclude that, for the networks under study, it has been quite worthwhile to use Proposition 1 as a preprocessing step before actually solving the various MAP problems. Because of the higher computational burden involved and its relative lack of additional value, the use of Proposition 2 has not been worthwhile for our networks and associated problems.

To conclude, in Section 3 we observed that fixing variables to their MAP value could serve to partition a MAP problem into smaller problems. Such a partition did not occur in our experiments. We would like to note, however, that we studied MAP problems without evidence only. We expect that in the presence of evidence MAP problems will more readily be partitioned into smaller problems.

## 5 Conclusions and discussion

The MAP problem for Bayesian networks is the problem of finding for a set of variables an instantiation of highest posterior probability given the available evidence. The problem has an unfavourable computational complexity, being $\mathrm{NP}^{\mathrm{PP}}$-hard in general. In this paper, we showed that computation of the marginal posterior probabilities of a variable $H$ given its Markov blanket may provide exact information about its value in a MAP solution. This information is valid for any MAP problem that includes the variable $H$. We further showed that computation of the marginal probabilities of $H$ given its MAP blanket may also provide exact information about its value. This information is valid, however, for a more restricted class of MAP problems. We argued that these results can be exploited for preprocessing MAP problems before they are actually solved using any state-of-the-art algorithm for this purpose.

We performed a preliminary experimental study of the practicability of the preprocessing steps by solving MAP problems for three different Bayesian networks using an exact branch-

Table 2: The number of network propagations without and with preprocessing using Propositions 1 and 2.

| | | | Prop. 1 | | Prop. 2 | | Prop. 1+2 | |
|---|---|---|---|---|---|---|---|---|
| *network* | *#MAP* | *#props.* | *#vars. f.* | *#props.* | *#vars. f.* | *#props.* (add.) | *#vars. f.* | *#props.* (add.) |
| WD | 5 | 6.0 | 1.2 | 4.8 | 0.0 | 6.0 (0.6) | $1.2 + 0.0$ | 4.8 (0.6) |
| | 10 | 11.0 | 2.4 | 8.6 | 1.0 | 10.0 (6.4) | $2.4 + 0.2$ | 8.4 (4.0) |
| CSF | 10 | 11.0 | 2.0 | 9.0 | 1.0 | 10.0 (1.6) | $2.0 + 0.4$ | 8.6 (0.4) |
| | 20 | 21.0 | 3.4 | 17.6 | 1.6 | 19.4 (8.2) | $3.4 + 0.6$ | 17.0 (5.0) |
| OESO+ | 17 | 24.8 | 3.4 | 18.4 | 0.8 | 22.0 (4.2) | $3.4 + 0.0$ | 18.4 (2.2) |
| | 34 | 49.8 | 5.8 | 40.8 | 1.8 | 47.4 (7.8) | $5.8 + 0.4$ | 40.0 (4.6) |

and-bound algorithm. As expected, the number of network propagations required by the algorithm is effectively decreased by fixing MAP variables to their appropriate values. We found that by building upon the concept of Markov blanket for 17.1% to 19.0% of the variables a MAP value could be fixed. Since the results of this preprocessing step are applicable to all future MAP computations and the computational burden involved thus is amortised, we considered it worthwhile to perform this step for the investigated networks. We would like to add that, because the computations involved are highly local, the step may also be feasibly applied to networks that are too large for the exact MAP algorithm used in the experiments. With respect to building upon the concept of MAP blanket, we found that for the investigated networks and associated MAP problems, the computations involved outweighed the reduction in network propagations that was achieved. Since the networks in our study were comparable with respect to size and connectivity, further experiments are necessary before any definitive conclusion can be drawn with respect to this preprocessing step.

In our further research, we will expand the experiments to networks with different numbers of variables, different cardinality and diverging connectivity. More specifically, we will investigate for which types of network preprocessing is most profitable. In our future experiments we will also take the effect of evidence into account. We will further investigate if the class of MAP problems that can be feasibly solved can be extended by our preprocessing method. We hope to report further results in the near future.

**Acknowledgements**

This research was partly supported by the Netherlands Organisation for Scientific Research (NWO).

## References

B.M.P. Aleman, L.C. van der Gaag and B.G. Taal. 2000. *A Decision Model for Oesophageal Cancer – Definition Document* (internal publication in Dutch).

P.L. Geenen and L.C. van der Gaag. 2005. Developing a Bayesian network for clinical diagnosis in veterinary medicine: from the individual to the herd. In *3rd Bayesian Modelling Applications Workshop*, held in conjunction with the 21st Conference on Uncertainty in Artificial Intelligence.

M. Korver and P.J.F. Lucas. 1993. Converting a rule-based expert system into a belief network. *Medical Informatics*, 18:219–241.

J. Park and A. Darwiche. 2001. Approximating MAP using local search. In *17th Conference on Uncertainty in Artificial Intelligence*, pages 403–410.

J. Park. 2002. MAP complexity results and approximation methods. In *18th Conference on Uncertainty in Artificial Intelligence*, pages 388–396.

J. Park and A. Darwiche. 2003. Solving MAP exactly using systematic search. In *19th Conference on Uncertainty in Artificial Intelligence*, pages 459–468.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Palo Alto.

# Quartet-Based Learning of Shallow Latent Variables

**Tao Chen and Nevin L. Zhang**
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology, Hong Kong, China
{csct,lzhang}@cse.ust.hk

**Abstract**

Hierarchical latent class(HLC) models are tree-structured Bayesian networks where leaf nodes are observed while internal nodes are hidden. We explore the following two-stage approach for learning HLC models: One first identifies the shallow latent variables – latent variables adjacent to observed variables – and then determines the structure among the shallow and possibly some other "deep" latent variables. This paper is concerned with the first stage. In earlier work, we have shown how shallow latent variables can be correctly identified from quartet submodels if one could learn them without errors. In reality, one does make errors when learning quartet submodels. In this paper, we study the probability of such errors and propose a method that can reliably identify shallow latent variables despite of the errors.

## 1 Introduction

Hierarchical latent class (HLC) models (Zhang, 2004) are tree-structured Bayesian networks where variables at leaf nodes are observed and hence are called *manifest variables (nodes)*, while variables at internal nodes are hidden and hence are called *latent variables (nodes)*. HLC models were first identified by Pearl (1988) as a potentially useful class of models and were first systematically studied by Zhang (2004) as a framework to alleviate the disadvantages of LC models for clustering. As a tool for cluster analysis, HLC Models produce more meaningful clusters than latent class models and they allow multi-way clustering at the same time. As a tool for probabilistic modeling, they can model high-order interactions among observed variables and help one to reveal interesting latent structures behind data. They also facilitate unsupervised profiling.

Several algorithms for learning HLC models have been proposed. Among them, the heuristic single hill-climbing (HSHC) algorithm developed by Zhang and Kočka (2004) is currently the most efficient. HSHC has been used to successfully analyze, among others, the CoIL Challenge 2000 data set (van der Putten and van Someren, 2004), which consists of 42 manifest variables and 5,822 records, and a data set about traditional Chinese medicine (TCM), which consists of 35 manifest variables and 2,600 records.

In terms of running time, HSHC took 98 hours to analyze the aforementioned TCM data set on a top-end PC, and 121 hours to analyze the CoIL Challenge 2000 data set. It is clear that HSHC will not be able to analyze data sets with hundreds of manifest variables.

Aimed at developing algorithms more efficient than currently available, we explore a two-stage approach where one (1) identifies the shallow latent variables, i.e. latent variables adjacent to observed variables, and (2) determines the structure among those shallow, and possibly some other "deep", latent variables. This paper is concerned with the first stage.

In earlier work (Chen and Zhang, 2005), we have shown how shallow latent variables can be correctly identified from quartet submodels if one could learn them without errors. In reality, one does make errors when learning quartet-submodels. In this paper, we study the probability of such errors and propose a method that

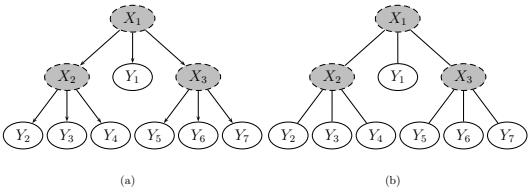



Figure 1: An example HLC model and the corresponding unrooted HLC model. The $X_i$'s are latent nodes and the $Y_j$'s are manifest nodes.

can reliably identify shallow latent variables despite of the errors.

## 2 HLC Models and Shallow Latent Variables

Figure 1 (a) shows an example HLC model. Zhang (2004) has proved that it is impossible to determine, from data, the orientation of edges in an HLC model. One can learn only *unrooted HLC models*, i.e. HLC models with all directions on the edges dropped. Figure 1 (b) shows an example unrooted HLC model. An unrooted HLC model represents a class of HLC models. Members of the class are obtained by rooting the model at various nodes. Semantically it is a Markov random field on an undirected tree. In the rest of this paper, we are concerned only with unrooted HLC models.

In this paper, we will use the term *HLC structure* to refer to the set of nodes in an HLC model and the connections among them. HLC structure is *regular* if it does not contain latent nodes of degree 2. Starting from an irregular HLC structure, we can obtain a regular structure by connecting the two neighbors of each latent node of degree 2 and then remove that node. This process is known as *regularization*. In this paper, we only consider regular HLC structures.

In an HLC model, a *shallow latent variable* (SLV) is one that is adjacent to at least one manifest variable. Two manifest variables are *siblings* if they are adjacent to the same (shallow) latent variable. For a given shallow latent variable $X$, all manifest variables adjacent to $X$ constitute a *sibling cluster*. In the HLC structure shown in Figure 1 (b), there are 3 sibling clusters, namely $\{Y_1\}$, $\{Y_2, Y_3, Y_4\}$, and $\{Y_5, Y_6, Y_7\}$. They correspond to the three latent variables in the model respectively.

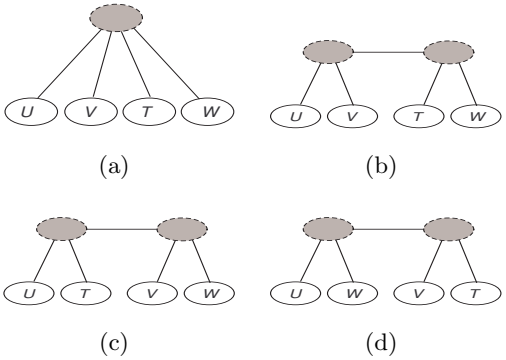



Figure 2: Four possible quartet substructures for a quartet $\mathbf{Q} = \{U, V, T, W\}$. The fork in (a) is denoted by $[UVTW]$, the dogbones in (b), (c), and (d) respectively by $[UV|TW]$, $[UT|VW]$, and $[UW|VT]$.

## 3 Quartet-Based SLV Discovery: The Principle

A shallow latent node is defined by its relationship with its manifest neighbors. Hence to identify the shallow latent nodes means to identify sibling clusters. To identify sibling clusters, we need to determine, for each pair of manifest variables $(U, V)$, whether $U$ and $V$ are siblings. In this section, we explain how to answer this question by inspecting quartet submodels.

A *quartet* is a set of four manifest variables, e.g., $\mathbf{Q} = \{U, V, T, W\}$. The *restriction* of an HLC model structure $\mathcal{S}$ onto $\mathbf{Q}$ is obtained from $\mathcal{S}$ by deleting all the nodes and edges not in the paths between any pair of variables in $\mathbf{Q}$. Applying regularization to the resulting HLC structure, we obtain the *quartet substructure* for $\mathbf{Q}$, which we denote by $\mathcal{S}|_{\mathbf{Q}}$. As shown in Figure 2, $\mathcal{S}|_{\mathbf{Q}}$ is either the *fork* $[UVTW]$, or one of the *dogbones* $[UV|TW]$, $[UT|VW]$, and $[UW|VT]$.

Consider the HLC structure in Figure 1 (b). The quartet substrcuture for $\{Y_1, Y_2, Y_3, Y_4\}$ is the fork $[Y_1Y_2Y_3Y_4]$, while that for $\{Y_1, Y_2, Y_4, Y_5\}$ is the dogbone $[Y_1Y_5|Y_2Y_4]$, and that for $\{Y_1, Y_2, Y_5, Y_6\}$ is the dogbone $[Y_1Y_2|Y_5Y_6]$.

It is obvious that if two manifest variables $U$ and $V$ are siblings in the structure $\mathcal{S}$, then they must be siblings in any quartet substructure that involves both of them. Chen and Zhang

(2005) has proved the converse. So, we have

**Theorem 1** *Suppose $\mathcal{S}$ is a regular HLC structure. Let $(U, V)$ be a pair of manifest variables. Then $U$ and $V$ are not siblings in $\mathcal{S}$ iff there exist two other manifest variables $T$ and $W$ such that $\mathcal{S}|_{\{U,V,T,W\}}$ is a dogbone where $U$ and $V$ are not siblings.*

Theorem 1 indicates that we can determine whether $U$ and $V$ are siblings by examining all possible quartets involving both $U$ and $V$. There are $\frac{(n-2)(n-3)}{2}$ such quartets, where $n$ is the number of manifest variables. We next present a result that allows us to do the same by examining only $n-3$ quartets.

Let $(U, V)$ be a pair of manifest variables and $T$ be a third one. We use $\mathcal{Q}_{UV|T}$ to denote the following collection of quartets:

$$\mathcal{Q}_{UV|T} := \{\{U, V, T, W\} | W \in \mathbf{Y} \backslash \{U, V, T\}\},$$

where $\mathbf{Y}$ is the set of all manifest variables. $T$ appears in every quartet in $\mathcal{Q}_{UV|T}$ and thus called a *standing member* of $\mathcal{Q}_{UV|T}$. It is clear that $\mathcal{Q}_{UV|T}$ consists of $n-3$ quartets. Chen and Zhang (2005) has also proved the following:

**Theorem 2** *Suppose $\mathcal{S}$ is a regular HLC structure. Let $(U, V)$ be a pair of manifest variables and $T$ be a third one. $U$ and $V$ are not siblings in $\mathcal{S}$ iff there exists a quartet $\mathbf{Q} \in \mathcal{Q}_{UV|T}$ such that $\mathcal{S}|_{\mathbf{Q}}$ is a dogbone where $U$ and $V$ are not siblings.*

In learning tasks, we do not know the structure of the generative model structure $\mathcal{S}$. Where do we obtain the quartet substructures? The answer is to learn them from data. Let $\mathcal{M}$ be an HLC model with a regular structure $\mathcal{S}$. Suppose that $\mathbf{D}$ is a collection of i.i.d samples drawn from $\mathcal{M}$. Each record in $\mathbf{D}$ contains values for the manifest variables, but not for the latent variables. Let $\texttt{QSL}(\mathbf{D}, \mathbf{Q})$ be a routine that takes data $\mathbf{D}$ and a quartet $\mathbf{Q}$ as inputs, and returns an HLC structure on the quartet $\mathbf{Q}$. One can use the HSHC algorithm to implement QSL, and one can first project the data $\mathbf{D}$ onto the quartet $\mathbf{Q}$ when learning the substructure for $\mathbf{Q}$.

Suppose QSL is error-free, i.e. $\texttt{QSL}(\mathbf{D}, \mathbf{Q}) = \mathcal{S}|_{\mathbf{Q}}$ for any quartet $\mathbf{Q}$. By Theorem 2, we can determine whether two manifest variables $U$ and $V$ are siblings (in the generative model) as follow:

- Pick a third manifest variable $T$.
- For each $\mathbf{Q} \in \mathcal{Q}_{UV|T}$, call $\texttt{QSL}(\mathbf{D}, \mathbf{Q})$.
- If $U$ and $V$ are not siblings in one of the resulting substructures, then conclude that they are not siblings (in the generative model).
- If $U$ and $V$ are siblings in all the resulting substructures, then conclude that they are siblings (in the generative model).

We can run the above procedure on each pair of manifest variables to determine whether they are siblings. Afterwards, we can summarize all the results using a *sibling graph*. The sibling graph is an undirected graph over the manifest variables where two variables $U$ and $V$ are adjacent iff they are determined as siblings.

If QSL is error-free, then each connected component of the sibling graph should be completely connected and correspond to one latent variable. For example, if the structure of the generative model is as Figure 3 (a), then the sibling graph that we obtain will be as Figure 3 (b). There are four completely connected components, namely $\{Y_1, Y_2, Y_3\}$, $\{Y_4, Y_5, Y_6\}$, $\{Y_7, Y_8, Y_9\}$, $\{Y_{10}, Y_{11}, Y_{12}\}$, which respectively correspond to the four latent variables in the generative structure.

## 4 Probability of Learning Quartet Submodels Correctly

In the previous section, we assumed that QSL is error-free. In reality, one does make mistakes when learning quartet submodels. We have empirically studied the probability of such errors.

For our experiments, QSL was implemented using the HSHC algorithm. For model selection, we tried each of the scoring functions, namely BIC (Schwarz, 1978), BICe (Kočka and Zhang, 2002), AIC (Akaike, 1974), and the Cheeseman-Stutz(CS) score (Cheeseman and Stutz, 1995).

We randomly generated around 20,000 quartet models. About half of them are fork-structured, while the rest are dogbone-structured. The cardinalities of the variables

range from 2 to 5. From each of the models, we sampled data sets of size 500, 1,000, 2,500, 5,000. The QSL was then used to analyze the data sets. In all the experiments, QSL produced either forks or dogbones. Consequently, there are only three classes of errors:

**F2D**: The generative model was a fork, and QSL produced a dogbone.

**D2F**: The generative model was a dogbone, and QSL produced a fork.

**D2D**: The generative model was a dogbone, and QSL produced a different dogbone.

The statistics are shown in Table 5. To understand the meaning of the numbers, consider the number 0.83% at the upper-left corner of the top table. It means that, when the sample size was 500, QSL returned dogbones in 0.83% percent, or 83, of the 10,011 cases where the generative models were forks. In all the other cases, QSL returned the correct fork structure.

It is clear from the tables that: The probability of D2F errors is large; the probability of F2D errors is small; and the probability of D2D errors is very small, especially when BIC or BICe are used for model selection. Also note that the probability of D2F decreases with sample size, but that of F2D errors do not. In the next section, we will use those observations when designing an algorithm for identifying shallow latent variables.

In terms of comparison among the scoring functions, BIC and BICe are clearly preferred over the other two as far as F2D and D2D errors are concerned. It is interesting to observe that BICe, although proposed as an improvement to BIC, is not as good as BIC when it comes to learning quartet models. For the rest of this paper, we use BIC.

## 5 Quartet-Based SLV Discovery: An Algorithm

The quartet-based approach to SLV discovery consists of three steps: (1) learn quartet submodels, (2) determine sibling relationships among manifest variables and hence obtain a sibling graph, and (3) introduce SLVs based on the sibling graph. Three questions ensue:

Table 1: Percentage of times that QSL produced the wrong quartet structure. The table on the top is for the fork-structured generative models, while the table at the bottom is for the dogbone-structured generative models.

| | 500(F2D) | 1000(F2D) | 2500(F2D) | 5000(F2D) |
|---|---|---|---|---|
| BIC | 0.83% | 1.87% | 3.70% | 3.93% |
| BICe | 1.42% | 3.16% | 6.58% | 7.86% |
| AIC | 12.8% | 10.2% | 6.81% | 4.85% |
| CS | 6.20% | 5.05% | 6.19% | 6.41% |
| Total=10011 | | | | |

| | 500 | | 1000 | | 2500 | | 5000 | |
|---|---|---|---|---|---|---|---|---|
| | D2F | D2D | D2F | D2D | D2F | D2D | D2F | D2D |
| BIC | 95.3% | 0.00% | 87.0% | 0.00% | 68.2% | 0.00% | 51.5% | 0.00% |
| BICe | 95.0% | 0.05% | 86.8% | 0.00% | 67.8% | 0.04% | 50.6% | 0.04% |
| AIC | 71.2% | 2.71% | 60.5% | 1.58% | 46.7% | 0.54% | 39.0% | 0.38% |
| CS | 88.5% | 1.93% | 83.4% | 0.74% | 67.0% | 0.30% | 51.5% | 0.13% |
| Total=10023 | | | | | | | | |

i) Which quartets should we use in Step 1?
ii) How do we determine sibling relationships in Step 2 based on results from Step 1?
iii) How do we introduce SLVs in Step 3 based on the sibling graph constructed in Step 2?

Our answer to the second question is simple: two manifest variables are regarded as non-siblings if they are not siblings in one of the quartet submodels. In the next two subsections, we discuss the other two questions.

### 5.1 SLV Introduction

As seen in Section 3, when QSL is error-free, the sibling graph one obtains has a nice property: every connected component is a fully connected subgraph. In this case, the rule for SLV introduction is obvious:

> Introduce one latent variable for each connected component.

When QSL is error-prone, the sibling graph one obtains no longer has the aforementioned property. Suppose data are sampled from a model with the structure shown in Figure 3 (a). Then what one obtains might be the graphs (c), (d), or (e) instead of (b).

Nonetheless, we still use the above SLV introduction rule for the general case. We choose it for its simplicity, and for the lack of better alternatives. This choice also endows the SLV discovery algorithm being developed with error tolerance abilities.

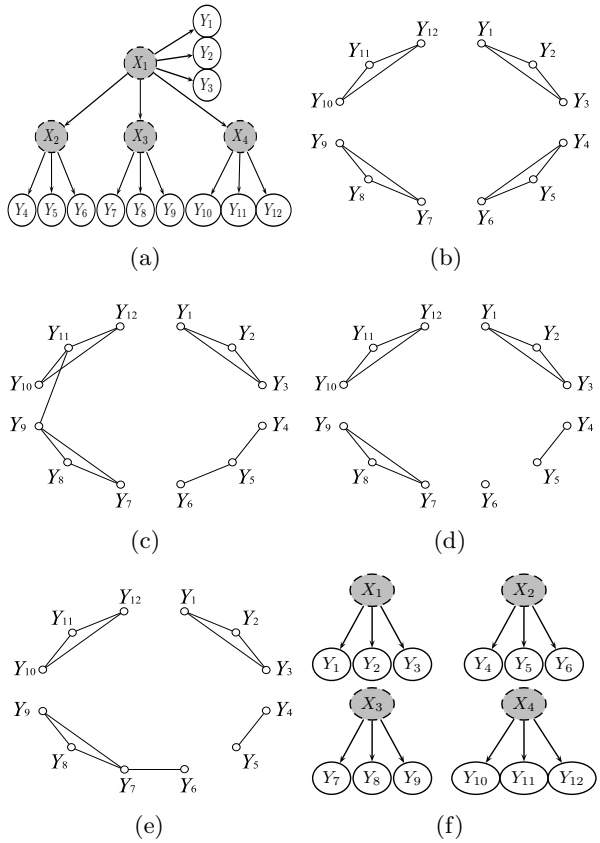



Figure 3: Generative model (a), sibling graphs (b, c, d, e), and shallow latent variables (f).

There are three types of mistakes that one can make when introducing SLVs, namely *latent omission, latent commission*, and *misclustering*. In the example shown in Figure 3, if we introduce SLVs based on the sibling graph (c), then we will introduce three latent variables. Two of them correspond respectively to the latent variables $X_1$ and $X_2$ in the generative model, while the third corresponds to a "merge" of $X_3$ and $X_4$. So, one latent variable is *omitted*.

If we introduce SLVs based on the sibling graph (d), then we will introduce five latent variables. Three of them correspond respectively to $X_1$, $X_3$, and $X_4$, while the other two are both related to $X_2$. So, one latent variable is *commissioned*.

If we introduce SLVs based on the sibling graph (e), then we will introduce four latent variables. Two of them correspond respectively to $X_1$ and $X_4$. The other two are related to $X_2$ and $X_3$, but there is not clear correspondence. This is a case of *misclustering*.

We next turn to Question 1. There, the most important concern is how to minimize errors.

### 5.2 Quartet Selection

To determine whether two manifest variables $U$ and $V$ are siblings, we can consider all quartets in $\mathcal{Q}_{UV|T}$, i.e. all the quartets with a third variable $T$ as a standing member. This selection of quartets will be referred to as the *parsimonious selection*. There are only $n-3$ quartets in the selection.

When `QSL` were error-free, one can use $\mathcal{Q}_{UV|T}$ to correctly determine whether $U$ and $V$ are siblings. As an example, suppose data are sampled from a model with the structure shown in Figure 3 (a), and we want to determine whether $Y_9$ and $Y_{11}$ are siblings based on data. Further suppose $Y_4$ is picked as the standing member. If `QSL` is error-free, we get 4 dogbones where $Y_9$ and $Y_{11}$ are not sibling, namely $[Y_9Y_7|Y_{11}Y_4]$, $[Y_9Y_8|Y_{11}Y_4]$, $[Y_9Y_4|Y_{11}Y_{10}]$, $[Y_9Y_4|Y_{11}Y_{12}]$, and hence conclude that $Y_9$ and $Y_{11}$ are not siblings.

In reality, `QSL` does make mistakes. According to Section 4, the probability of D2F errors is quite high. There is therefore good chance that, instead of the aforementioned 4 dogbones, we get 4 forks. In that case, $Y_9$ and $Y_{11}$ will be regarded as siblings, resulting in a *fake edge* in the sibling graph.

$\mathcal{Q}_{UV|T}$ represents one extreme when it comes to quartet selection. The other extreme is to use all the quartets that involve both $U$ and $V$. This selection of quartets will be referred to as the *generous selection*. Suppose $U$ and $V$ are non-siblings in the generative model. This generous choice will reduce the effects of D2F errors. As a matter of fact, there are now many more quartets, when compared with the case of parsimonious quartet selection, for which the true structures are dogbones with $U$ and $V$ being non-siblings. If we learn one of those structures correctly, we will be able to correctly identify $U$ and $V$ as non-siblings.

The generous selection also comes with a drawback. In our running example, consider the task of determining whether $Y_1$ and $Y_2$ are siblings based on data. There are 36 quartets

that involve $Y_1$ and $Y_2$ and for which the true structures are forks. Those include $[Y_1Y_2Y_4Y_7]$, $[Y_1Y_2Y_4Y_{10}]$, and so on. According to the Section 4, the probability for QSL to make an F2D mistake on any one of those quartets is small. But there are 36 of them. There is good chance for QSL to make an F2D mistake on one of them. If the structure QSL learned for $[Y_1Y_2Y_4Y_7]$, for instance, turns out to be the dogbone $[Y_1Y_4|Y_2Y_7]$, then $Y_1$ and $Y_2$ will be regarded as non-siblings, and hence the edge between $Y_1$ and $Y_2$ will be *missing* from the sibling graph.

Those discussions point to the middle ground between parsimonious and generous quartet selection. One natural way to explore this middle ground is to use several standing members instead of one.

### 5.3 The Algorithm

Figure 4 shows an algorithm for discovering shallow latent variables, namely DiscoverSLVs. The algorithm first calls a subroutine ConstructSGraph to construct a sibling graph, and then finds all the connected components of the graph. It is understood that one latent variable is introduced for each of the connected components.

The subroutine ConstructSGraph starts from the complete graph. For each pair of manifest variables $U$ and $V$, it considers all the quartets that involve $U$, $V$, and one of the $m$ standing members $T_i$ $(i = 1, 2, \ldots, m)$. QSL is called to learn a submodel structure for each of the quartets. If $U$ and $V$ are not siblings in one of the quartet substructures, the edge between $U$ and $V$ is deleted.

DiscoverSLVs has error tolerance mechanism naturally built in. This is mainly because it regards connected components in sibling graph as sibling clusters. Let $\mathbf{C}$ be a sibling cluster in the generative model. The vertices in $\mathbf{C}$ will be placed in one cluster by DiscoverSLVs if they are in the same connected component in the sibling graph $G$ produced by ConstructSGraph. It is not required for variables in $\mathbf{C}$ to be pairwise connected in $G$. Therefore, a few mistakes by ConstructSGraph when determining sibling relationships are tolerated. When the set $\mathbf{C}$ is not very small, it takes a few or more mistakes in the right combination to break up $\mathbf{C}$.

Algorithm DiscoverSLVs($\mathbf{D}, m$):

1. $G \leftarrow$ ConstructSGraph($\mathbf{D}, m$).
2. **return**
   the list of the connected components of $G$.

Algorithm ConstructSGraph($\mathbf{D}, m$):

1. $G \leftarrow$ complete graph over manifest nodes $\mathbf{Y}$.
2. **for** each edge $(U, V)$ of $G$,
3. pick $\{T_1, \cdots, T_m\} \subseteq \mathbf{Y} \backslash \{U, V\}$
4. **for** each $\mathbf{Q} \in \cup_{i=1}^{m} \mathcal{Q}_{UV|T_i}$,
5. **if** QSL($\mathbf{D}, \mathbf{Q}$)=$[U * |V *]$
6. delete edge $(U, V)$ from $G$, break.
7. **endFor**.
8. **endFor**.
9. **return** $G$.

Figure 4: An algorithm for learning SLVs.

## 6 Empirical Evaluation

We have carried out simulation experiments to evaluate the ability of DiscoverSLVs in discovering latent variables. This section describes the setup of the experiments and reports our findings.

The generative models in the experiments share the same structure. The structure consists of 39 manifest variables and 13 latent variables. The latent variables form a complete 3-ary tree of height two. Each latent variable in the structure is connected to 3 manifest variables. Hence all latent variables are shallow. The cardinalities of all variables were set at 3.

We created 10 generative models from the structure by randomly assigning parameter values. From each of the 10 generative models, we sampled 5 data sets of 500, 1,000, 2,500, 5,000 and 10,000 records. DiscoverSLVs was run on each of the data sets three times, with the number of standing members $m$ set at 1, 3 and 5 respectively. The algorithms were implemented in Java and all experiments were run on a Pentium 4 PC with a clock rate of 3.2 GHz.

The performance statistics are summarized in Table 2. They consist of errors at three different

Table 2: Performance statistics of our SLV discovery algorithm.

| | QSL-level | | | Edge-level | | SLV-level | | |
|---|---|---|---|---|---|---|---|---|
| m | D2F | D2D | F2D | ME | FE | LO | LCO | MC |
| Sample size = 500 | | | | | | | | |
| 1 | 77.0%(4.5%) | 0.06%(0.05%) | 0.40%(0.13%) | 1.0(1.1) | 88.2(9.2) | 11(3.0) | 0(0) | 0(0) |
| 3 | 71.8%(4.0%) | 0.05%(0.03%) | 0.36%(0.14%) | 2.9(1.3) | 28.0(8.3) | 9.3(3.1) | 0(0) | 0.1(0.3) |
| 5 | 68.8%(3.0%) | 0.05%(0.02%) | 0.30%(0.14%) | 3.3(1.6) | 14.7(6.0) | 5.1(1.9) | 0.2(0.4) | 0(0) |
| Sample size = 1000 | | | | | | | | |
| 1 | 65.3%(6.2%) | 0.01%(0.03%) | 0.17%(0.14%) | 0.3(0.6) | 55.3(15.7) | 11.2(0.7) | 0(0) | 0(0) |
| 3 | 58.3%(2.7%) | 0.02%(0.02%) | 0.10%(0.07%) | 0.7(0.9) | 10.1(6.5) | 4.8(2.9) | 0(0) | 0(0) |
| 5 | 56.6%(3.2%) | 0.01%(0.01%) | 0.10%(0.08%) | 1.1(0.7) | 6.2(3.8) | 2.3(1.3) | 0.1(0.3) | 0(0) |
| Sample size = 2500 | | | | | | | | |
| 1 | 50.1%(2.3%) | 0.00%(0.00%) | 0.04%(0.07%) | 0.2(0.4) | 20.4(8.9) | 8.6(2.3) | 0(0) | 0(0) |
| 3 | 38.0%(4.6%) | 0.00%(0.00%) | 0.02%(0.04)% | 0.2(0.4) | 3.1(3.7) | 1.4(1.4) | 0(0) | 0(0) |
| 5 | 37.3%(3.8%) | 0.00%(0.01%) | 0.07%(0.08)% | 1.1(1.4) | 1.3(2.5) | 1.5(0.9) | 0.1(0.3) | 0(0) |
| Sample size = 5000 | | | | | | | | |
| 1 | 32.6%(5.5%) | 0.00%(0.00%) | 0.03%(0.09%) | 0(0) | 7.7(6.0) | 3.9(2.4) | 0(0) | 0(0) |
| 3 | 25.2%(4.4%) | 0.01%(0.01%) | 0.04%(0.06%) | 0.3(0.5) | 1.5(2.7) | 0.5(0.7) | 0(0) | 0(0) |
| 5 | 25.7%(3.9%) | 0.00%(0.01%) | 0.10%(0.11%) | 0.9(0.9) | 0.9(2.7) | 0.1(0.3) | 0(0) | 0(0) |
| Sample size = 10000 | | | | | | | | |
| 1 | 21.8%(5.9%) | 0.00%(0.00%) | 0.02%(0.07%) | 0(0) | 2.0(3.3) | 1.2(1.8) | 0(0) | 0(0) |
| 3 | 17.5%(3.9%) | 0.00%(0.00%) | 0.05%(0.06%) | 0.2(0.4) | 0.4(1.2) | 0.1(0.3) | 0(0) | 0(0) |
| 5 | 17.4%(4.1%) | 0.00%(0.01%) | 0.03%(0.04%) | 0.6(0.8) | 0.1(0.3) | 0.1(0.3) | 0(0) | 0(0) |

levels of the algorithm: the errors made when learning quartet substructures (QSL-level), the errors made when determining sibling relationships between manifest variables (edge-level), and the errors made when introducing SLVs (SLV-level). Each number in the table is an average over the 10 generative models. The corresponding standard deviations are given in parentheses.

**QSL-level errors**: We see that the probabilities of the QSL-level errors are significantly smaller than those reported in Section 4. This is because we deal with strong dependency models here, while the numbers in Section 4 are about general models. This indicates that strong dependency assumption does make learning easier. On the other hand, the trends remain the same: the probability of D2F errors is large, that of F2D errors is small, and that of D2D errors are very small. Moreover, the probability of D2F errors decreases with sample size.

**Edge-level errors**: For the edge-level, the numbers of missing edges (ME) and the number of fake edges (FE) are reported. We see that the number of missing edges is always small, and in general it increases with the number of standing members $m$. This is expected since the larger $m$ is, the more quartets one examines, and hence the more likely one makes F2D errors.

The number of fake edges is large when the sample size is small and $m$ is small. In general, it decreases with sample size and $m$. It dropped to 1.3 when for the case of sample size 2,500 and $m = 5$. This is also expected. As $m$ increases, the number of quartets examined also increases. For two manifest variables $U$ and $V$ that are not siblings in the generative model, the probability of obtaining a dogbone (despite D2F errors) where $U$ and $V$ are not siblings also increases. The number of fake edges decreases with $m$ because as $m$ increases, the probability of D2F errors decreases.

**SLV-level errors**: We now turn to SLV-level errors. Because there were not many missing edges, true sibling clusters of the generative models were almost never broken up. There are only five exceptions. The first exception happened for one generative model in the case of sample size 500 and $m = 3$. In that case, a manifest variable from one true cluster was placed into another, resulting in one misclustering error (MC).

The other four exceptions happened for the following combinations of sample size and $m$: (500, 5), (1000, 5), (2500, 5). In those cases, one true sibling cluster was broken into two clusters, resulting in four latent commission errors (LCO).

Fake edges cause clusters to merge and hence lead to latent omission errors. In our experiments, the true clusters were almost never broken up. Hence a good way to measure latent omission errors is to use the total number of shallow latent variables, i.e. 13, minus the number of clusters returned by `DiscoverSLVs`. We call this the number of LO errors. In Table 2,we

see that the number of LO errors decreases with sample size and the number of standing members $m$. It dropped to 1.4 when for the case of sample size 2,500 and $m = 3$. When the sample size was increased to 10,000 and $m$ set to 3 or 5, LO errors occurred only once for one of the 10 generative models.

**Running time**: The running times of `DiscoverSLVs` are summarized in the following table (in hours). For the sake of comparison, we also include the times HSHC took attempting to reconstruct the generative models based on the same data as used by `DiscoverSLVs`. We see that `DiscoverSLVs` took only a small fraction of the time that HSHC took, especially in the cases with large samples. This indicates that the two-stage approach that we are exploring can result algorithms significantly more efficient than HSHC.

| RunningTime(hrs) | 500 | 1000 | 2500 | 5000 | 10000 |
|---|---|---|---|---|---|
| $m$=1 | 1.05 | 1.06 | 0.92 | 0.89 | 0.84 |
| $m$=3 | 1.57 | 1.52 | 1.49 | 1.79 | 1.91 |
| $m$=5 | 1.85 | 2.07 | 2.17 | 2.72 | 3.02 |
| HSHC | 4.65 | 8.69 | 23.7 | 43.0 | 118.6 |

The numbers of quartets examined by `DiscoverSLVs` are summarized in the following table. We see that `DiscoverSLVs` examined only a small fraction of all the $C_{39}^4$=82251 possible quartets. We also see that doubling $m$ does not imply doubling the number of quartets examined, nor doubling the running time. Moreover, the number of quartets examined decreases with sample size. This is because the probability of D2F errors decrease with sample size.

| | 500 | 1000 | 2500 | 5000 | 10000 |
|---|---|---|---|---|---|
| $m$=1 | 5552 | 4191 | 2861 | 2131 | 1816 |
| $m$=3 | 8326 | 5939 | 4659 | 4292 | 4112 |
| $m$=5 | 9801 | 8178 | 6776 | 6536 | 6496 |
| Total: 82251 | | | | | |

## 7 Related Work

Linear latent variable graphs (LLVGs) are a special class of structural equation models. Variables in such models are continuous. Some are observed, while others are latent. Sliva *et al.* (2003) has studied the problem of identifying SLVs in LLVGs. Their approach is based on the Tetrad constraints (Spirtes *et al.*, 2000), and its complexity is $O(n^6)$.

Phylogenetic trees (PT) (St. John *et al.*, 2003) can be viewed as a special class of HLC models. The quartet-based method for PT reconstruction first learn submodels for all $C_n^4$ possible quartets and then use them to build the overall tree ( St. John *et al.*, 2003).

## 8 Conclusions

We have empirically studied the probability of making errors when learning quartet HLC models and have observed some interesting regularities. In particular, we observed the probability of D2F errors is high and decreases with sample size, while the probability of F2D errors is low. Based on those observations, we have developed an algorithm for discovering shallow latent variables in HLC models reliably and efficiently.

**Acknowledgements**

Research on this work was supported by Hong Kong Grants Council Grant #622105.

## References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automotic Control,*.

Cheeseman, P. and Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. In *Advances in knowledge discovery and data mining.*

Chen, T. and Zhang, N.L. (2006). Quartet-Based Learning of HLC Models: Discovery of Shallow Latent Variables. In *AIMath-06.*

Kočka, T. and Zhang, N.L. (2002). Dimension correction for hierarchical latent class models. In *UAI-02.*

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.*

Schwarz, G. (1978). Estimating the dimension of a model. *Annuals of Statistics.*

Silva, R., Scheines, R., Glymour, C. and Spirtes, P. (2003). Learning measurement models for unobserved variables. In *UAI-03.*

Spirtes, P., Glymour, C. and Scheines, R. (2000). *Causation, Prediction and Search.*

St. John, K., Warnow, T., Moret, B.M.E. and Vawter, L. (2003) Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining. *Journal of Algorithms.*

van der Putten, P. and van Someren, M. (2004). A Bias-Variance Analysis of a Real World Learning Problem: The CoIL Challenge 2000. *Machine Learning.*

Zhang, N.L. (2004). Hierarchical latent class models for cluster analysis. *JMLR.*

Zhang, N.L. and Kočka, T. (2004). Efficient learning of hierarchical latent class models. In *ICTAI-04.*

# Continuous Decision MTE Influence Diagrams

Barry R. Cobb
*cobbbr@vmi.edu*
Department of Economics and Business
Virginia Military Institute
Lexington, VA, 24450 U.S.A.

**Abstract**

Mixtures of truncated exponentials (MTE) influence diagrams can represent decision problems with discrete decision variables without limitations on the distributions of continuous chance variables or the nature of the utility functions. This paper presents an extension of MTE influence diagrams that allows both discrete and continuous decision variables. This new model—the continuous decision MTE influence diagram—develops decision rules for continuous decision variables as a function of their continuous parents. In this model, a continuous decision node is marginalized by replacing it with a deterministic chance node and applying an operation derived from the method of convolutions in probability theory. In experiments, continuous decision MTE influence diagrams provide an increase in maximum expected utility when compared to existing methods.

## 1 Introduction

An *influence diagram* is a compact graphical representation for a decision problem under uncertainty. Initially, influence diagrams were proposed by Howard and Matheson (1984) as a front-end for decision trees. Subsequently, Olmsted (1983) and Shachter (1986) developed methods for evaluating an influence diagram directly without converting it to a decision tree. These methods assume that all uncertain variables in the model are represented by discrete probability mass functions (PMF's) and that decision variables have discrete state spaces.

Influence diagram models that permit continuous chance and decision variables include Gaussian influence diagrams (Shachter and Kenley, 1989), mixtures of Gaussians influence diagrams (Poland and Shachter, 1993), and linear-quadratic conditional Gaussian influence diagrams (Madsen and Jensen, 2005). Each of these models exploits multivariate normal probability theory to provide the decision strategy and expected utility that solves the influence diagram.

Cobb and Shenoy (2004) introduce mixtures of truncated exponentials (MTE) influence diagrams, which are influence diagrams where probability distributions and utility functions are represented by MTE potentials. Decision variables in MTE influence diagrams must have discrete state spaces; however, many decision problems in practice have continuous decision variables.

Consider the following decision problem from economics (see Figure 1):

> A monopolist faces uncertain demand dependent on favorable ($M = 1$) or unfavorable ($M = 0$) market conditions. Demand is reflected in the price ($P$) it receives for its output ($Q$). The monopolist cannot directly observe demand, but rather relies on the results ($R$) of a market survey to gauge demand, and thus predict price ($P$). Based on the survey results ($R$), the monopolist must determine its optimal output ($Q$).

The monopolist can produce from 0 to 70 units and has the following utility function:

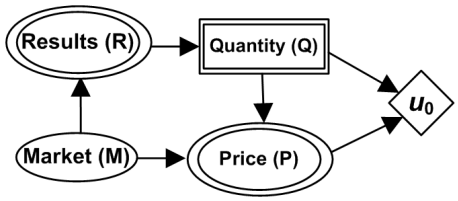



Figure 1: An influence diagram for the monopolist's decision problem.

$u_0(p,q) = (p-5)\cdot q - 200000$. The probability of an unfavorable market ($M=0$) is 0.40. The mean of the conditional Gaussian probability density function (PDF) for price ($P$) is a linear function of quantity ($Q$) produced, which is based on the market conditions as follows: $P \mid \{Q, M=0\} \sim N(5000-70q, 100^2)$ and $P \mid \{Q, M=1\} \sim N(10000-50q, 1000^2)$. Test results ($R$) are given as a percentage of consumers surveyed who intend to buy the product and are modeled by the following beta PDF's: $R \mid M=0 \sim Beta(1.3, 2.7)$ and $R \mid M=1 \sim Beta(2.7, 1.3)$.

Influence diagrams with continuous decision variables and non-Gaussian continuous chance variables are difficult to solve using current methodology. This paper introduces the continuous decision MTE influence diagram (CDMTEID), a model which allows any combination of discrete and continuous chance variables with no restrictions on the type of probability distribution, as well as discrete and/or continuous decision variables. CDMTEID's can also accommodate conditionally deterministic chance variables. An operation derived from the method of convolutions in probability theory is used to eliminate a continuous decision node during the solution phase in a CDMTEID.

The remainder of this paper is organized as follows. Section 2 gives notation and definitions. Section 3 describes operations used to solve CDMTEID's. Section 4 presents the CDMTEID solution to the example problem. Section 5 provides conclusions. This paper is extracted from a longer working paper (Cobb, 2006).

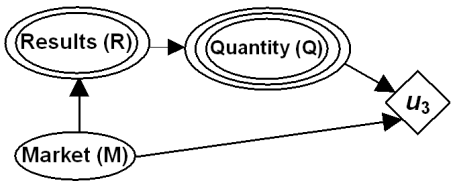



Figure 2: Influence diagram for Example 1.

## 2 Notation and Definitions

### 2.1 Notation

Variables will be denoted by capital letters, e.g., $A, B, C$. Sets of variables will be denoted by boldface capital letters, $\mathbf{Y}$ if all are discrete chance variables, $\mathbf{Z}$ if all are continuous chance variables, $\mathbf{D}$ if all are decision variables, or $\mathbf{X}$ if the components are a mixture of discrete chance, continuous chance, and decision variables. If $\mathbf{X}$ is a set of variables, $\mathbf{x}$ is a configuration of specific states of those variables. The discrete, continuous, or mixed state space of $\mathbf{X}$ is denoted by $\Omega_{\mathbf{X}}$.

MTE probability potentials, discrete probability potentials, and deterministic potentials are denoted by lower-case greek letters, e.g., $\alpha$, $\beta$, $\gamma$. MTE utility potentials are denoted by $u_i$.

In graphical representations, decision variables are represented by rectangular nodes (with a single border for discrete and a double border for continuous), chance variables are represented by ovals (with a single border for discrete, a double border for continuous, and a triple border if the chance variable is deterministic), and utility functions are represented by diamonds.

**Example 1.** Consider the influence diagram in Figure 2. In this model, $M$ is a discrete variable which can take on values $M=0$ or $M=1$. The variables $R$ and $Q$ are continuous chance variables, with $Q$ a deterministic chance variable ($Q$ is conditionally deterministic given a value of $R$). The utility function ($u_3$) is a continuous function of $Q$.

### 2.2 Mixtures of Truncated Exponentials

A mixture of truncated exponentials (MTE) potential in an influence diagram has the following

definition, which is a modification of the definition proposed by Rumí and Salmerón (2005).

*MTE potential.* Let $\mathbf{X}$ be a mixed $n$-dimensional variable. Let $\mathbf{Y} = (Y_1, \ldots, Y_d)$, $\mathbf{Z} = (Z_1, \ldots, Z_c)$, and $\mathbf{D} = (D_1, \ldots, D_f)$ be the discrete chance, continuous chance, and decision variable parts of $\mathbf{X}$, respectively, with $c + d + f = n$. A function $\phi : \Omega_{\mathbf{X}} \mapsto \mathcal{R}^+$ is an MTE potential if one of the next three conditions holds:

1. $\mathbf{Y} \cup \mathbf{D} = \emptyset$ and $\phi$ can be written as

$$\phi(\mathbf{x}) = \phi(\mathbf{z}) = a_0 + \sum_{i=1}^{m} a_i \exp\Big\{ \sum_{j=1}^{c} b_i^{(j)} z_j \Big\} \tag{1}$$

   for all $\mathbf{x} \in \Omega_{\mathbf{X}}$, where $a_i, i = 0, \ldots, m$ and $b_i^{(j)}$, $i = 1, \ldots, m$, $j = 1, \ldots, c$ are real numbers.

2. $\mathbf{Y} \cup \mathbf{D} = \emptyset$ and there is a partition $\Omega_1, \ldots, \Omega_k$ of $\Omega_{\mathbf{Z}}$ into hypercubes such that $\phi$ is defined as

$$\phi(\mathbf{x}) = \phi_h(\mathbf{x}) \qquad \text{if } \mathbf{x} \in \Omega_h, \tag{2}$$

   where each $\phi_h, h = 1, ..., k$ can be written in the form of equation (1) (i.e. each $\phi_h$ is an MTE potential on $\Omega_h$).

3. $\mathbf{Y} \cup \mathbf{D} \neq \emptyset$ and for each fixed value $(\mathbf{y}, \mathbf{d}) \in \Omega_{\mathbf{Y} \cup \mathbf{D}}$, $\phi_{\mathbf{y},\mathbf{d}}(\mathbf{z})$ can be defined as in (2).

In the definition above, $k$ is the number of *pieces*, and $m$ is the number of exponential *terms* in each piece of the MTE potential. In the third case, the potential *fragments* $\phi_{\mathbf{y},\mathbf{d}}(\mathbf{z})$ for all $(\mathbf{y}, \mathbf{d}) \in \Omega_{\mathbf{Y},\mathbf{D}}$ constitute the MTE potential for $\{\mathbf{Y}, \mathbf{Z}, \mathbf{D}\}$. In this paper, all MTE probability and utility potentials are equal to zero in unspecified regions. In CDMTEID's, all probability distributions and utility functions are approximated by MTE potentials.

The definition presented here assumes that decision variables are discrete. This paper presents a method for developing a decision rule for a continuous decision variable as a function of its continuous parents; however, the MTE representation of this method first uses a discrete approximation to the continuous decision variable.

### 2.3 MTE Probability Densities

Suppose $\phi'$ is an input MTE potential for $\mathbf{X} = \mathbf{Y} \cup \mathbf{Z} \cup \mathbf{D}$ representing a PDF for $Z \in \mathbf{Z}$ given its parents $\mathbf{X} \setminus \{Z\}$. If we can verify that

$$\int_{\Omega_Z} \phi'(\mathbf{x}, z)\, dz = 1\ , \tag{3}$$

for all $\mathbf{x} \in \Omega_{\mathbf{X} \setminus Z}$, we state that $\phi'$ is an MTE density for $Z$. We assume that all input MTE probability potentials in a CDMTEID are normalized prior to the solution phase.

### 2.4 Deterministic Potential

A *deterministic potential* describes the linear deterministic relationship between a set of variables $\mathbf{Z} = \{Z_1, \ldots, Z_c\}$. A deterministic potential (Cobb and Shenoy, 2005) for $\mathbf{Z}$ is defined as an equation

$$[g(\mathbf{z}) = 0] = \{w_p \cdot [g_p(\mathbf{z}) = 0]\}_{p=1}^{P}\ , \tag{4}$$

where $w_p$, $p = 1, \ldots, P$ are constants. The equation $g_p(\mathbf{z}) = 0$ defines a linear deterministic relationship, where $g_p(\mathbf{z}) = a_{p1} z_1 + \cdots + a_{pc} z_c + b_p$, and where $a_{p1}, \ldots, a_{pc}$ and $b_p$ are real numbers. The coefficient $a_{pi}$ on $Z_i \in \mathbf{Z}$ is equal to 1 when the deterministic potential is specified as a conditional potential for $Z_i$ given $\mathbf{Z} \setminus Z_i$. The *factors* $w_p \cdot [g_p(\mathbf{z}) = 0]$ are maintained as a decomposed set of weighted equations, with $w_p$ representing the *weights* for all $p = 1, \ldots P$. The weights $(w_p)$ typically result from the marginalization of a discrete variable, as shown in Example 3 in Section 3.2.1.

We term $[g(\mathbf{z}) = 0]$ a potential in the sense that any variable $Z_i$ takes on the value $z_i = (-a_{p1} z_1 - \cdots - a_{p,i-1} z_{i-1} - a_{p,i+1} z_{i+1} - \cdots - a_{pc} z_c - b_p)\,/a_{pi}$ with probability 1 and any other value with probability 0. As with MTE potentials, multiple fragments of the form in (4) can define a deterministic potential for a set of variables $\mathbf{X}$.

The fragments can be parameterized by either sets of discrete chance and decision variables, or by partitions of hypercubes of continuous chance variables, as shown below.

**Example 2.** In the influence diagram of Figure 2, $Q$ is conditionally deterministic given $R$. This relationship is represented by the deterministic potential fragment $[g_0(q,r) = 0]$ or $[q - 46.667r - 30.112 = 0]$ if $0 \leq r \leq 0.8547$, and by the deterministic potential fragment $[g_1(q,r) = 0]$ or $[q - 0r - 70 = 0]$ if $0.8547 < r \leq 1$.

## 3 Solving CDMTEID's

CDMTEID's are solved by using the fusion algorithm developed by Shenoy (1993) and the operations presented in this section. The fusion algorithm involves deleting variables from the network in a sequence which respects the information constraints (represented by arcs pointing to decision variables in influence diagrams) in the problem. This condition ensures that unobserved chance variables are deleted before decision variables. The fusion method applies to problems where there is only one utility function (or a joint utility function which factors multiplicatively into several utility potentials).

Combination of two MTE potentials is pointwise multiplication. Marginalization of chance variables from MTE potentials is integration over continuous chance variables being removed and summation over discrete chance variables being removed. Details of these two operations can be found in (Cobb and Shenoy, 2004). Other combination and marginalization operations required for CDMTEID's are described below.

### 3.1 Combination of MTE and Deterministic Potentials

Let $\phi_{\mathbf{y},\mathbf{d}}(\mathbf{z}_1)$ be an MTE probability potential on $\mathbf{X} = \mathbf{Y} \cup \mathbf{Z} \cup \mathbf{D}$ and let $[g_{\mathbf{y},\mathbf{d}}(\mathbf{z}_2) = 0]$ be a deterministic potential on $\mathbf{X} = \mathbf{Y} \cup \mathbf{Z} \cup \mathbf{D}$, where $\mathbf{Z} = \mathbf{Z}_1 \cup \mathbf{Z}_2$. The combination of $\phi_{\mathbf{y},\mathbf{d}}$ and $g_{\mathbf{y},\mathbf{d}}$ is a potential $\zeta$ for $\mathbf{X}$ defined as

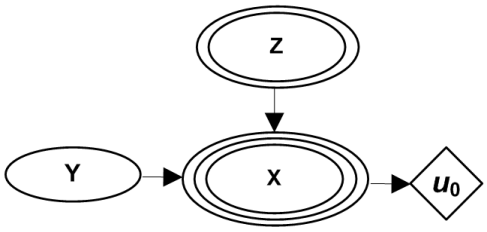



Figure 3: Influence diagram for Example 3.

$$\begin{aligned}\zeta_{\mathbf{y},\mathbf{d}}(\mathbf{z}) &= (\phi_{\mathbf{y},\mathbf{d}} \otimes g_{\mathbf{y},\mathbf{d}})_{\mathbf{y},\mathbf{d}}(\mathbf{z}) \\ &= (\{\phi_{\mathbf{y},\mathbf{d}}(\mathbf{z}_1), [g_{\mathbf{y},\mathbf{d}}(\mathbf{z}_2) = 0]\}) \ ,\end{aligned} \tag{5}$$

for all $\mathbf{z} \in \Omega_{\mathbf{Z}}$. According to the LAZY propagation scheme (Madsen and Jensen, 1999) the potentials are not combined, but rather maintained as a decomposed set of potentials during combination.

### 3.2 Marginalization

#### 3.2.1 Discrete Chance Variables from MTE and Deterministic Potentials

This operation is illustrated by example (see (Cobb, 2006) for a formal definition).

**Example 3.** Consider the influence diagram in Figure 3, where $P(Y = 0) = 0.5$ and $P(Y = 1) = 0.5$. The variable $X$ is conditionally deterministic given $\{Y, Z\}$, a relationship represented by the deterministic potential fragments $g_0(x, z, Y = 0) = [x - 2z + 1 = 0]$ and $g_1(x, z, Y = 1) = [x - 0.25z - 1 = 0]$. The removal of $Y$ from the combination of $g$ for $\{X, Y, Z\}$ and the PMF for $Y$ results in the following deterministic potential: $\{0.5 \cdot [x - 2z + 1 = 0], 0.5 \cdot [x - 0.25z - 1 = 0]\}$.

The values for the deterministic potential are weighted with the probability values upon removal of the discrete variable.

#### 3.2.2 Continuous Chance Variables from MTE and Deterministic Potentials

Marginalization of a continuous variable $Z_i$ from the combination of an MTE potential and a deterministic potential is substitution of the inverse of the equation(s) in the deterministic

potential into the MTE potential. This operation is derived from the method of convolutions in probability theory, as explained in (Cobb and Shenoy, 2005).

Let $\phi_{\mathbf{y},\mathbf{d}}(\mathbf{z}_1)$ be an MTE potential on $\mathbf{X}_1 = \mathbf{Y} \cup \mathbf{Z}_1 \cup \mathbf{D}$ and let $[g_{\mathbf{y},\mathbf{d}}(\mathbf{z}_2) = 0]$ be a deterministic potential on $\mathbf{X}_2 = \mathbf{Y} \cup \mathbf{Z}_2 \cup \mathbf{D}$. Assume $\mathbf{Z} = \mathbf{Z}_1 \cup \mathbf{Z}_2$, $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$, and that each $[g_{\mathbf{y},\mathbf{d},p}(\mathbf{z}_2) = 0]$, $p = 1, \ldots, P$, is invertible in $Z_i \in (\mathbf{Z}_1 \cap \mathbf{Z}_2)$. The marginal of $\{\phi_{\mathbf{y},\mathbf{d}}, g_{\mathbf{y},\mathbf{d}}\}$ for a set of variables $\mathbf{X}' = \mathbf{Y} \cup (\mathbf{Z} \setminus Z_i) \cup \mathbf{D} \subseteq \mathbf{X}$ is computed as

$$\begin{aligned} &(\phi_{\mathbf{y},\mathbf{d}} \otimes g_{\mathbf{y},\mathbf{d}})^{\downarrow \mathbf{X}'}_{\mathbf{y},\mathbf{d}}(\mathbf{z}') \\ &= \sum_{p=1}^{P} K_p \cdot w_p \cdot \phi_{\mathbf{y},\mathbf{d}}\left(h_{\mathbf{y},\mathbf{d},p}\left(\mathbf{z}'_2\right), \mathbf{z}'_1\right) \quad , \end{aligned} \tag{6}$$

for all $\mathbf{x}' \in \Omega_{\mathbf{X}'}$ where $\mathbf{z} = (\mathbf{z}', z_i)$, $\mathbf{z}_1 = (\mathbf{z}'_1, z_i)$, $\mathbf{z}_2 = (\mathbf{z}'_2, z_i)$, and

$$\begin{aligned} &h_{\mathbf{y},\mathbf{d},p}(\mathbf{z}'_2) = (-a_{p1} z_1 - \cdots \\ &-a_{p,i-1} z_{i-1} - a_{p,i+1} z_{i+1} - \cdots - a_{pc} z_c - b_p) / a_{pi} \cdot \end{aligned}$$

The constant $K_p$ is $\frac{1}{\mid a_{pi} \mid}$ if $\phi_{\mathbf{y},\mathbf{d}}$ is an MTE probability potential and 1 if $\phi_{\mathbf{y},\mathbf{d}}$ is an MTE utility potential, where $a_{pi}$ represents the coefficient on variable $Z_i$ in $g_{\mathbf{y},\mathbf{d},p}(\mathbf{z}_2)$.

### 3.2.3 Discrete Decision Variables

Marginalization with respect to a discrete decision variable is only defined for MTE utility potentials. Let $u$ be an MTE utility potential for $\mathbf{X} = \mathbf{Y} \cup \mathbf{Z} \cup \mathbf{D}$, where $D \in \mathbf{D}$. The marginal of $u$ for a set of variables $\mathbf{X} - \{D\}$ is an MTE utility potential computed as

$$u^{\downarrow(\mathbf{X}-\{D\})}(\mathbf{y}, \mathbf{z}, \mathbf{d}') = \max_{d \in \Omega_D} u(\mathbf{y}, \mathbf{z}, \mathbf{d}) \tag{7}$$

for all $(\mathbf{y}, \mathbf{z}, \mathbf{d}') \in \Omega_{\mathbf{X}-\{D\}}$ where $\mathbf{d} = (\mathbf{d}', d)$. Marginalization of decision variables results in an MTE potential. For details, see Cobb and Shenoy (2004).

### 3.2.4 Continuous Decision Variables

Eliminating a continuous decision variable from a CDMTEID is a three-step process:

Step 1: Create a discrete approximation to the continuous decision variable and apply the procedure in Section 3.2.3.

Step 2: Using least squares regression, create a decision rule for the continuous decision variable as a function of its continuous parent(s).

Step 3: Construct a deterministic potential from the decision rule developed in Step 2, convert the continuous decision variable to a deterministic chance variable, and marginalize the variable using the procedure in Section 3.2.2.

This operation will be described for the case where a decision variable has one continuous parent, but multivariate regression can be used to extend the operation for the case where a decision variable has multiple continuous parents. Suppose we want to eliminate a continuous decision variable $D$ with continuous parent $Z$ from the influence diagram using the fusion algorithm. Let $u$ be an MTE utility potential for $\{Z, D\}$. First, a $v$-point discrete approximation is created for $D$. Each qualitative state of the discrete approximation corresponds with a real numbered value in the continuous state space of the variable, $\Omega_D = \{d : d_{min} \leq d \leq d_{max}\}$. The real numbered values associated with the qualitative states are determined as $d_t = d_{min} + (t - 0.5) \cdot (d_{max} - d_{min}) / v$ for $t = 1, \ldots, v$. For simplicity, the qualitative states will also be referred to as $d_1, \ldots, d_v$.

Marginalization of $D$ from the utility potential $u$ involves finding the value of $d_t$ that maximizes $u$ for each point in the continuous domain of $Z$. The following $k$-piece component MTE potential for $Z$ is derived from the discretized continuous decision variable $D$ using the procedure in Section 3.2.3:

$$u_{max}(z) = \begin{cases} u_1(z) & \text{if } e_0 \leq z < e_1 \\ u_2(z) & \text{if } e_1 \leq z < e_2 \\ \vdots & \\ u_k(z) & \text{if } e_{k-1} \leq z \leq e_k \end{cases}$$

where $e_{j-1}$ and $e_j$ are the lower and upper bounds of the domain of the variable $Z$ in $j$-

th piece of the MTE utility potential $u_{max}$, as determined by the procedure in Section 3.2.3. These bounds are defined such that $(e_{i-1}, e_i) \cap (e_{j-1}, e_j) = \emptyset$ for all $i, j = 1, \ldots, k$, $i \neq j$ and $\bigcup_{j=1}^{k} (e_{j-1}, e_j) = (z_{min}, z_{max})$, where the state space of $Z$ is given as $\Omega_Z = \{z : z_{min} \leq z \leq z_{max}\}$. Let $\Psi(z) = \{d^{(1)}, ..., d^{(k)}\}$ be a vector of real-numbered values $d_t$ of the decision variable $D$ that correspond with $u_1, \ldots, u_k$ in $u_{max}$, i.e. $\Psi$ is the decision rule obtained upon the removal of $D$. The decision rule is a function where $\Psi(z) = d^{(j)}$ if $e_{j-1} \leq z < e_j$ for all $j = 1, \ldots, k$.

The decision rule $\Psi$ is transformed to a decision rule $\hat{\Psi}$ that is stated as a function of $Z$. Using least squares regression, we obtain a linear equation of the form $f(z) = b_0 + b_1 z$. The values $\mathbf{b} = [b_0, b_1]$ are calculated as $\mathbf{b} = \left(\Delta^\top \Delta\right)^{-1} \Delta^\top \theta$ where $\Theta = [d^{(1)}, d^{(2)}, \ldots, d^{(k)}]^\top$ and

$$\Delta = \begin{bmatrix} 1 & \frac{e_0+e_1}{2} \\ 1 & \frac{e_1+e_2}{2} \\ \vdots & \vdots \\ 1 & \frac{e_{k-1}+e_k}{2} \end{bmatrix} .$$

The decision rule is then stated as

$$\hat{\Psi}(z) = \begin{cases} d_{min} & \text{if } b_0 + b_1 z < d_{min} \\ b_0 + b_1 z & \\ & \text{if } d_{min} \leq b_0 + b_1 z \leq d_{max} \\ d_{max} & \text{if } b_0 + b_1 z > d_{max} \ , \end{cases}$$

where the first and third regions of $\hat{\Psi}(z)$ are added to ensure $D$ takes on a value within its state space. The decision rule above is used to construct a deterministic potential $[g(z, d) = 0]$. Variable $D$ has been converted to a deterministic chance variable and is eliminated from the model by using the operation described in Section 3.2.2.

The class of MTE potentials is closed under each of the operations required to solve MTE influence diagrams (Cobb and Shenoy, 2004) and the convolution operation used to eliminate continuous decision variables (Cobb and Shenoy, 2005).

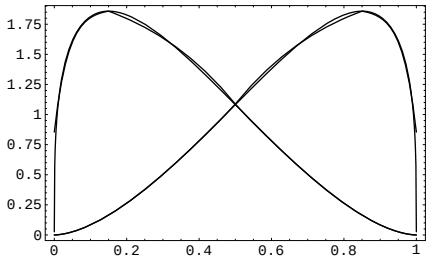



Figure 4: The MTE potential fragments which constitute the MTE potential $\beta$ for $\{R, M\}$.

## 4 Example

This section presents the CDMTEID solution to the monopolist's decision problem. Additional numerical details of MTE potentials can be found in (Cobb, 2006).

### 4.1 Representation

The MTE potential for $P$ given $\{M, Q\}$ is a 2-piece MTE approximation to the normal PDF (Cobb and Shenoy 2006a) with $\mu = 5000 - 70q$ and $\sigma^2 = 100^2$ given $M = 0$, and $\mu = 10000 - 50q$ and $\sigma^2 = 1000^2$ given $M = 1$. The potential $\gamma$ for $\{P, M, Q\}$ has potential fragments $\gamma(p, q, M = 0)$ and $\gamma(p, q, M = 1)$.

The MTE approximation to the utility function $u_0$ for $\{P, Q\}$ is denoted by $u_1$ and is determined using the procedure described in (Cobb and Shenoy, 2004). MTE approximations to the beta PDF's for $R$ given $M$ are constructed using the procedure in (Cobb *et al.*, 2006). These MTE potential fragments—which constitute the MTE potential $\beta$ for $\{R, M\}$—are displayed graphically in Figure 4 overlayed on the corresponding beta PDF's. The PMF for $M$ is represented by the potential $\alpha$ where $\alpha(0) = P(M = 0) = 0.4$ and $\alpha(1) = P(M = 1) = 0.6$.

Although $Q$ is a continuous decision variable, we will initially use a three-point ($v = 3$) discrete approximation with $q_1 = 11.667$, $q_2 = 35$, and $q_3 = 58.333$ to apply the solution procedure.

### 4.2 Solution

The CDMTEID solution proceeds by eliminating the variables in the sequence $P$, $M$, $Q$, $R$.

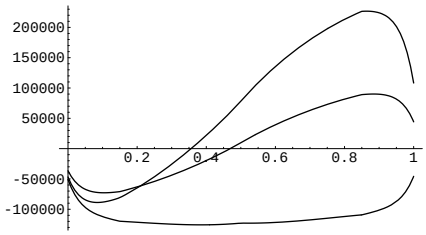



Figure 5: The utility potential fragments which constitute the utility potential $u_3$.

To remove $P$, we calculate $u_2 = (u_1 \otimes \gamma)^{\downarrow\{M,Q\}}$. The values for $u_2$ are calculated as

$$u_2(q, M = 0) = \int_{\Omega_P} u_1(p, q) \cdot \gamma(p, q, M = 0)\ dp\ ,$$
$$u_2(q, M = 1) = \int_{\Omega_P} u_1(p, q) \cdot \gamma(p, q, M = 1)\ dp\ .$$

To remove $M$, we calculate $u_3 = (u_2 \otimes \alpha \otimes \beta)^{\downarrow\{Q,R\}}$. The utility potential fragments $u_3(r, Q = q_1)$, $u_3(r, Q = q_2)$, and $u_3(r, Q = q_3)$ are shown graphically in Figure 5. Removing $Q$ involves first finding $\text{Max}\{u_3(r, Q = q_1), u_3(r, Q = q_2), u_3(r, Q = q_3)\}$ at each point in the domain of $R$. In this case, we find $u_3(r, Q = q_2) \approx u_3(r, Q = q_3)$ at $R = 0.2095$. Next, the decision rule is developed by using this value.

### 4.2.1 Creating a Decision Rule for a Continuous Decision Variable

Using the procedure in Section 3.2.3, we determine the optimal strategy of producing $Q = q_2 = 35$ if $0 < r < 0.2095$, and $Q = q_3 = 58.333$ if $0.2095 \leq r < 1$. Using least squares regression (see Section 3.2.4), the following decision rule is obtained:

$$\hat{\Psi}(r) = \begin{cases} 46.667 \cdot r + 30.112 & \text{if } 0 \leq r \leq 0.8547 \\ 70 & \text{if } 0.8547 < r \leq 1 \end{cases} .$$

To convert $Q$ to a deterministic chance variable, we use $\hat{\Psi}(r)$ to define the deterministic potential $g$ for $\{Q, R\}$ as in Example 2 of Section 2.4 (the domain of $R$ precludes the possibility of producing zero units).

Table 1: Decision strategy for an CDMTEID solution with eight states for $Q$.

| Test Results ($R$) | Quantity Produced ($Q$) |
|---|---|
| $0 \leq r \leq 0.1655$ | Q=39.375 |
| $0.1655 < r \leq 0.2895$ | Q=48.125 |
| $0.2895 < r \leq 0.3975$ | Q=56.875 |
| $0.3975 < r \leq 1$ | Q=65.625 |

A CDMTEID solution with eight discrete states for the decision variable $Q$ yields the decision rule in Table 1 (some states of $Q$ are not optimal for any values of $R$). Define $\Theta = [39.375, 48.125, 56.875, 65.625]^\top$ and

$$\Delta = \begin{bmatrix} 1 & \frac{0+0.1655}{2} \\ 1 & \frac{0.1655+0.2895}{2} \\ 1 & \frac{0.2895+0.3975}{2} \\ 1 & \frac{0.3975+1}{2} \end{bmatrix} = \begin{bmatrix} 1 & 0.08275 \\ 1 & 0.2275 \\ 1 & 0.3435 \\ 1 & 0.69875 \end{bmatrix} .$$

The resulting decision rule is based on the linear equation $f(r) = 41.403r + 38.501$ and the deterministic potential $[g(r, q) = 0]$ for $\{Q, R\}$ has fragments $[g_0(r, q) = 0]$ or $[q - 41.403r - 38.501 = 0]$ if $0 \leq r \leq 0.7608$ and $[g_1(r, q) = 0]$ or $[q - 0r - 70 = 0]$ if $0.7608 < r \leq 1$.

### 4.2.2 Converting the Decision Variable to a Deterministic Chance Variable

To eliminate the decision variable $Q$, we consider the revised influence diagram in Figure 2 that replaces the decision node with a deterministic probability node. The potentials remaining after $Q$ is converted to a deterministic chance variable are $u_3$ for $\{Q, R\}$ and $g$ for $\{Q, R\}$. To marginalize a continuous chance variable from the combination of a utility potential and a deterministic potential, we substitute an expression obtained from the deterministic potential for the variable to be removed into the utility potential, as detailed in Section 3.2.2. In this case, a new MTE utility potential is:

$$u_4(r) = \begin{cases} u_3(46.667 \cdot r + 30.112, r) & \\ \qquad\qquad \text{if } 0 \leq r \leq 0.8547 & \\ u_3(70, r) \qquad \text{if } 0.8547 < r \leq 1 \end{cases} .$$

Integrating $u_4$ over the domain of $R$ gives the final expected utility of 83729. Creating the decision rule specified by the function $\hat{\Psi}(r)$ yields an increase in expected value of 9210 over the decision rule created using an MTE influence diagram with a discrete decision variable. Using the same procedure, the maximum expected utility for the monopolist using a CDMTEID with eight discrete states for $Q$ is 89686, which is 3053 higher than an MTE influence diagram with eight states for $Q$. Cobb (2006) gives a detailed comparison of the CDMTEID solution to similar MTE and discrete influence diagrams.

## 5 Conclusions

Continuous decision MTE influence diagrams—which determine a decision rule for a continuous decision variable as a function of its continuous parents—show potential to improve decision-making at a lower computational cost than discrete influence diagrams or MTE influence diagrams. Previous methods for handling continuous decision variables in influence diagrams require either normal distributions for continuous chance variables and/or discrete approximations to continuous decision variables. The continuous decision MTE influence diagram allows a more precise decision rule and the ability to exploit the continuous nature of test results.

### Acknowledgments

The author is grateful for the insightful comments of three anonymous reviewers which significantly improved the paper.

## References

Cobb, B.R. (2006), "Refining decision rules in influence diagrams," Working Paper, Virginia Military Institute. Available for download at: `www.vmi.edu/media/ecbu/cobb/CONTID.pdf`.

Cobb, B.R. and P.P. Shenoy (2004), "Hybrid influence diagrams using mixtures of truncated exponentials," in M. Chickering and J. Halpern (eds.), *Uncertainty in Artificial Intelligence*, 20, 85–93.

Cobb, B.R. and P.P. Shenoy (2005), "Hybrid Bayesian networks with linear deterministic variables," in F. Bacchus and T. Jaakkola (eds.), *Uncertainty in Artificial Intelligence*, 21, 136–144.

Cobb, B.R. and P.P. Shenoy (2006a), "Inference in hybrid Bayesian networks using mixtures of truncated exponentials," *International Journal of Approximate Reasoning*, 41(3), 257–286.

Cobb, B.R., Shenoy, P.P. and R. Rumí (2006), "Approximating probability density functions in hybrid Bayesian networks with mixtures of truncated exponentials," *Statistics and Computing*, 16(3), 293–308.

Howard, R.A. and J.E. Matheson (1984), "Influence diagrams," in R.A. Howard and J.E. Matheson (eds.), *Readings on the Principles and Applications of Decision Analysis*, 2, 719–762, Strategic Decisions Group, Menlo Park, CA.

Madsen, A.L. and F. Jensen (2005), "Solving linear-quadratic conditional Gaussian influence diagrams," *International Journal of Approximate Reasoning*, 38(3), 263–282.

Madsen, A.L. and F.V. Jensen (1999), "Lazy evaluation of symmetric Bayesian decision problems," in K.B. Laskey and H. Prade (eds.), *Uncertainty in Artificial Intelligence*, 15, 382–390.

Olmsted, S.M. (1983), "On representing and solving decision problems," Ph.D. Thesis, Department of Engineering–Economic Systems, Stanford University, Stanford, CA.

Poland, W.B. and R.D. Shachter (1993), "Mixtures of Gaussians and minimum relative entropy techniques for modeling continuous uncertainties," in D. Heckerman and E.H. Mamdani (eds.), *Uncertainty in Artificial Intelligence*, 9, 183–190.

Rumí, R. and A. Salmerón (2005), "Penniless propagation with mixtures of truncated exponentials," in L. Godo (ed.), *Symbolic and Quantitative Approaches to Reasoning under Uncertainty*, Lecture Notes in Artificial Intelligence, 3571, 39–50.

Shachter, R.D. (1986), "Evaluating influence diagrams," *Operations Research*, 34(6), 871–882.

Shachter, R.D. and C.R. Kenley (1989), "Gaussian influence diagrams," *Management Science*, 35(5), 527–550.

Shenoy, P.P. (1993), "A new method for representing and solving Bayesian decision problems," in D.J. Hand (ed.), *Artificial Intelligence Frontiers in Statistics: AI and Statistics III*, Chapman and Hall, London, 119–138.

# Discriminative Scoring of Bayesian Network Classifiers: a Comparative Study

Ad Feelders and Jevgenijs Ivanovs
Department of Information and Computing Science
Universiteit Utrecht, The Netherlands

**Abstract**

We consider the problem of scoring Bayesian Network Classifiers (BNCs) on the basis of the conditional loglikelihood (CLL). Currently, optimization is usually performed in BN parameter space, but for perfect graphs (such as Naive Bayes, TANs and FANs) a mapping to an equivalent Logistic Regression (LR) model is possible, and optimization can be performed in LR parameter space. We perform an empirical comparison of the efficiency of scoring in BN parameter space, and in LR parameter space using two different mappings. For each parameterization, we study two popular optimization methods: conjugate gradient, and BFGS. Efficiency of scoring is compared on simulated data and data sets from the UCI Machine Learning repository.

## 1 Introduction

Discriminative learning of Bayesian Network Classifiers (BNCs) has received considerable attention recently (Greiner et al., 2005; Pernkopf and Bilmes, 2005; Roos et al., 2005; Santafé et al., 2005). In discriminative learning, one chooses the parameter values that maximize the *conditional* likelihood of the class label given the attributes, rather then the *joint* likelihood of the class label and the attributes. It is well known that conditional loglikelihood (CLL) optimization, although arguably more appropriate in a classification setting, is computationally more expensive because there is no closed-form solution for the ML estimates and therefore numerical optimization techniques have to be applied. Since in structure learning of BNCs many models have to be scored, the efficiency of scoring a single model is of considerable interest.

For BNCs with perfect independence graphs (such as Naive Bayes, TANs, FANs) a mapping to an equivalent Logistic Regression (LR) model is possible, and optimization can be performed in LR parameter space. We consider two such mappings: one proposed in (Roos et al., 2005), and a different mapping, that, although relatively straightforward, has to our knowledge not been proposed before in discriminative learning of BNCs. We conjecture that scoring models in LR space using our proposed mapping is more efficient than scoring in BN space, because the logistic regression model has fewer parameters than its BNC counterpart and because the LR model is known to have a strictly concave loglikelihood function. To test this hypothesis we perform experiments to compare the efficiency of model fitting with both LR parameterizations, and the more commonly used BN parameterization.

This paper is structured as follows. In section 2 we introduce the required notation and basic concepts. Next, in section 3 we describe two mappings from BNCs with perfect graphs to equivalent LR models. In section 4 we give a short description of the optimization methods used in the experiments, and motivate their choice. Subsequently, we compare the efficiency of discriminative learning in LR parameter space and BN parameter space, using the optimization methods discussed. Finally, we present the conclusions in section 7.

## 2 Preliminaries

### 2.1 Bayesian Networks

We use uppercase letters for random variables and lowercase for their values. Vectors are written in boldface. A Bayesian network (BN) $(\mathbf{X}, G = (V, E), \boldsymbol{\theta})$ consists of a discrete random vector $\mathbf{X} = (X_0, \ldots, X_n)$, a directed acyclic graph (DAG) $G$ representing the directed independence graph of $\mathbf{X}$,

and a set of conditional probabilities (parameters) $\boldsymbol{\theta}$. $V = \{0, 1, \ldots, n\}$ is the set of nodes of $G$, and $E$ the set of directed edges. Node $i$ in $G$ corresponds to random variable $X_i$. With $pa(i)$ ($ch(i)$) we denote the set of parents (children) of node $i$ in $G$. We write $\mathbf{X}_S, S \subseteq \{0, \ldots, n\}$ to denote the projection of random vector $\mathbf{X}$ on components with index in $S$. The parameter set $\boldsymbol{\theta}$ consists of the conditional probabilities

$$\theta_{x_i|\mathbf{x}_{pa(i)}} = P(X_i = x_i|\mathbf{X}_{pa(i)} = \mathbf{x}_{pa(i)}), 0 \leq i \leq n$$

We use $\mathcal{X}_i = \{0, \ldots, d_i - 1\}$ to denote the set of possible values of $X_i$, $0 \leq i \leq n$. The set of possible values of random vector $\mathbf{X}_S$ is denoted $\mathcal{X}_S = \times_{i \in S} \mathcal{X}_i$. We also use $\mathcal{X}_i^- = \mathcal{X}_i \setminus \{0\}$, and likewise $\mathcal{X}_S^- = \times_{i \in S} \mathcal{X}_i^-$

In a BN classifier there is one distinguished variable called the class variable; the remaining variables are called attributes. We use $X_0$ to denote the class variable; $X_1, \ldots, X_n$ are the attributes. To denote the attributes, we also write $\mathbf{X}_A$, where $A = \{1, \ldots, n\}$. We define $\pi(i) = pa(i) \backslash \{0\}$, the non-class parents of node $i$, and $\phi(i) = \{i\} \cup \pi(i)$.

Finally, we recall the definition of a *perfect* graph: a directed graph in which all nodes that have a common child are connected is called *perfect*.

### 2.2 Logistic Regression

The basic assumption of logistic regression (Anderson, 1982) for binary class variable $X_0 \in \{0, 1\}$ is

$$\ln \frac{P(X_0 = 1|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} = w_0 + \sum_{i=1}^{k} w_i Z_i, \qquad (1)$$

where the predictors $Z_i$ ($i = 1, \ldots, k$) can be single attributes from $\mathbf{X}_A$, but also functions of one or more attributes from $\mathbf{X}_A$. In words: the log posterior odds are linear in the parameters, not necessarily in the basic attributes.

Generalization to a non-binary class variable $X_0 \in \mathcal{X}_0$ gives

$$\ln \frac{P(X_0 = x_0|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} = w_0^{(x_0)} + \sum_{i=1}^{k} w_i^{(x_0)} Z_i, \quad (2)$$

for all $x_0 \in \mathcal{X}_0^-$. This model is often referred to as the multinomial logit model or polychotomous logistic regression model.

It is well known that the loglikelihood function of the logistic regression model is concave and has unique maximum (provided the data matrix $Z$ is of full column rank) attained for finite $\mathbf{w}$ except in two special circumstances described in (Anderson, 1982).

### 2.3 Log-linear models

Let $G = (V, E)$ be the (undirected) independence graph of random vector $\mathbf{X}$, that is $E$ is the set of edges $(i, j)$ such that whenever $(i, j)$ is not in $E$, the variables $X_i$ and $X_j$ are independent conditionally on the rest. The log-linear expansion of a graphical log-linear model is

$$\ln P(\mathbf{x}) = \sum_{C \subseteq V} u_C(\mathbf{x}_C)$$

where the sum is taken over all complete subgraphs $C$ of $G$, and all $\mathbf{x}_C \in \mathcal{X}_C^-$, that is, $u_C(\mathbf{x}_C) = 0$ for $i \in C$ and $x_i = 0$ (to avoid overparameterization). The $u$-term $u_\varnothing(\mathbf{x})$ is just a constant. It is well known that for BN's with perfect directed independence graph, an equivalent graphical log-linear model is obtained by simply dropping the direction of the edges.

## 3 Mapping to Logistic Regression

In this section we discuss two different mappings from BNCs with a perfect independence graph to equivalent logistic regression models. Equivalent here means that, assuming $P(\mathbf{X}) > 0$, the BNC and corresponding LR model, represent the same set of conditional distributions of the class variable.

### 3.1 Mapping of Roos et al.

Roos et al. (Roos et al., 2005) define a mapping from BNCs whose canonical form is a perfect graph, to equivalent LR models. The canonical form is obtained by (1) taking the Markov blanket of $X_0$ (2) marrying any unmarried parents of $X_0$. This operation does clearly not change the conditional distribution of $X_0$. They show that if this canonical form is a perfect graph, then the BNC can be mapped to an equivalent LR model. Their mapping creates an LR model with predictors (and corresponding parameters) as follows

1. $Z_{\mathbf{x}_{pa(0)}} = I(\mathbf{X}_{pa(0)} = \mathbf{x}_{pa(0)})$ with parameter $w^{(x_0)}_{\mathbf{x}_{pa(0)}}$, for $\mathbf{x}_{pa(0)} \in \mathcal{X}_{pa(0)}$.
2. $Z_{\mathbf{x}_{\phi(i)}} = I(\mathbf{X}_{\phi(i)} = \mathbf{x}_{\phi(i)})$ with parameter $w^{(x_0)}_{\mathbf{x}_{\phi(i)}}$, for $i \in ch(0)$ and $\mathbf{x}_{\phi(i)} \in \mathcal{X}_{\phi(i)}$.

For a given BNC with parameter value $\boldsymbol{\theta}$ an equivalent LR model is obtained by putting

$$w^{(x_0)}_{\mathbf{x}_{pa(0)}} = \ln \theta_{x_0|\mathbf{x}_{pa(0)}}, \quad w^{(x_0)}_{\mathbf{x}_{\phi(i)}} = \ln \theta_{x_i|\mathbf{x}_{pa(i)}}$$

### 3.2 Proposed mapping

Like in the previous section, we start from the canonical graph which is assumed to be perfect. Hence, we obtain an equivalent graphical log-linear model by simply dropping the direction of the edges. We then have

$$\begin{aligned}
&\ln \frac{P(X_0 = x_0|\mathbf{X}_A)}{P(X_0 = 0|\mathbf{X}_A)} \\
&= \ln \frac{P(X_0 = x_0, \mathbf{X}_A)/P(\mathbf{X}_A)}{P(X_0 = 0, \mathbf{X}_A)/P(\mathbf{X}_A)} \\
&= \ln P(X_0 = x_0, \mathbf{X}_A) - \ln P(X_0 = 0, \mathbf{X}_A),
\end{aligned}$$

for $x_0 \in \mathcal{X}_0^-$. Filling in the log-linear expansion for $\ln P(X_0 = x_0, \mathbf{X}_A)$ and $\ln P(X_0 = 0, \mathbf{X}_A)$, we see immediately that $u$-terms that do not contain $X_0$ cancel, and furthermore that $u$-terms with $X_0 = 0$ are constrained to be zero by our identification restrictions. Hence we get

$$\begin{aligned}
&\ln P(X_0 = x_0, \mathbf{x}_A) - \ln P(X_0 = 0, \mathbf{x}_A) = \\
&\quad u_{\{0\}}(X_0 = x_0) + \sum_C u_C(X_0 = x_0, \mathbf{x}_C) = \\
&\qquad\qquad w^{(x_0)}_{\varnothing} + \sum_C w^{(x_0)}_{\mathbf{x}_C}
\end{aligned}$$

where $C$ is any complete subgraph of $G$ not containing $X_0$. Hence, to map to a LR model, we create variables

$$I(\mathbf{X}_C = \mathbf{x}_C), \qquad \mathbf{x}_C \in \mathcal{X}_C^-$$

to obtain the LR specification

$$\ln \frac{P(X_0 = x_0|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} = w^{(x_0)}_{\varnothing} + \sum_C w^{(x_0)}_{\mathbf{x}_C} I(\mathbf{X}_C = \mathbf{x}_C)$$

This LR specification models the same set of conditional distributions of $X_0$ as the corresponding undirected graphical model, see for example (Sutton and McCallum, 2006).

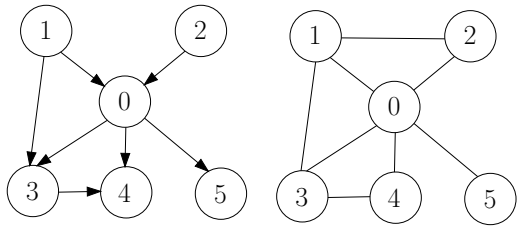



Figure 1: Example BNC (left); undirected graph with same conditional distribution of class (right).

### 3.3 Example

Consider the BNC depicted in figure 1. Assuming all variables are binary, and using this fact to simplify notation, this maps to the equivalent LR model (with 9 parameters)

$$\begin{aligned}
\ln \frac{P(X_0 = 1|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} = w_{\varnothing} + w_{\{1\}}X_1 + w_{\{2\}}X_2 + \\
w_{\{3\}}X_3 + w_{\{4\}}X_4 + w_{\{5\}}X_5 + \\
w_{\{1,2\}}X_1X_2 + w_{\{1,3\}}X_1X_3 + w_{\{3,4\}}X_3X_4
\end{aligned}$$

In the parameterization of (Roos et al., 2005), we map to the equivalent LR_Roos model (with 14 parameters)

$$\begin{aligned}
\ln \frac{P(X_0 = 1|\mathbf{Z})}{P(X_0 = 0|\mathbf{Z})} = w_{\{1,2\}=(0,0)}Z_{\{1,2\}=(0,0)} + \\
w_{\{1,2\}=(0,1)}Z_{\{1,2\}=(0,1)} + w_{\{1,2\}=(1,0)}Z_{\{1,2\}=(1,0)} + \\
w_{\{1,2\}=(1,1)}Z_{\{1,2\}=(1,1)} + w_{\{1,3\}=(0,0)}Z_{\{1,3\}=(0,0)} + \\
\ldots + w_{\{3,4\}=(1,1)}Z_{\{3,4\}=(1,1)} + \\
w_{\{5\}=(0)}Z_{\{5\}=(0)} + w_{\{5\}=(1)}Z_{\{5\}=(1)}
\end{aligned}$$

In both cases we set $\mathbf{w}^{(0)} = \mathbf{0}$.

## 4 Optimization methods

In the experiments, we use two optimization methods: conjugate gradient(CG) and variable metric (BFGS) algorithms (Nash, 1990). Conjugate Gradient is commonly used for discriminative learning of BN parameters, see for example (Greiner et al., 2005; Pernkopf and Bilmes, 2005). In a study of Minka (Minka, 2001), it was shown that CG and BFGS are efficient optimization methods for the logistic regression task.

BFGS is a Hessian-based algorithm, which updates an approximate inverse Hessian matrix of size

$r^2$ at each step, where $r$ is the number of parameters. CG on the other hand works with a vector of size $r$. This obviously makes each iteration less costly, however, in general CG exhibits slower convergence in terms of iterations.

Both algorithms at step $k$ compute an update direction $\mathbf{u}_{(k)}$, followed by a line search. This is a one-dimensional search, which looks for a step size $\alpha$ maximizing $f(\alpha) = CLL(\mathbf{w}_{(k)} + \alpha\mathbf{u}_{(k)})$, where $\mathbf{w}_{(k)}$ is a vector of parameter values at step $k$. It is argued in (Nash, 1990) that neither algorithm benefits from too large a step being taken. Even more, for BFGS it is not desirable that the increase in the function value is different in magnitude from the one determined by the gradient value, $(\mathbf{w}_{(k+1)} - \mathbf{w}_{(k)})^T\mathbf{g}_{(k)}$. Therefore, simple *acceptable point* search is suggested for both methods. In case of CG an additional step is made. Once an acceptable point has been found, we have sufficient information to fit a parabola to the projection of the function on the search direction. The parabola requires three pieces of information: the function value at the end of the last iteration (or the initial point), the projection of the gradient at this point onto the search direction, and the new function value at the acceptable point. If the CLL value at the maximum of the parabola is larger than the CLL value at the acceptable point, then the former becomes the starting point for the next iteration. Another approach to CG line search is Brent's line search method (Press et al., 1992). It iteratively fits a parabola to 3 points. The main difference with the previous approach is that we do find an optimum in the given update direction. This, however, requires more function evaluations at each iteration.

In our experiments we use the implementation of the CG and BFGS methods of the `optim` function of `R` (Venables and Ripley, 2002) which is based on the source code from (Nash, 1990). In addition we implemented CG with Brent's line search (with relative precision for Brent's method set to 0.0002). We refer to this algorithm as CGB. The conjugate gradient method may use different heuristic formulas for computing the update direction. Our preliminary study showed that the difference in performance between these heuristics is small. We use the Polak-Ribiere formula, suggested in (Greiner et al., 2005) for optimization in the BN parameter space.

In our study we compare the rates of convergence for 9 optimization techniques: 3 optimization algorithms (CG, CGB, BFGS) used in 3 parameter spaces (LR, LR_Roos, BN). We compare convergence in terms of (1) iterations of the update direction calculation, and (2) floating point operations (flops). The number of flops provides a fair comparison of the performance of the different methods, but the number of iterations gives us additional insight into the behavior of the methods.

Let $cost_{CLL}$ and $cost_{grad}$ denote the costs in flops of CLL and gradient evaluations respectively, and let $count_{CLL}$ be the number of times CLL is evaluated in a particular iteration. We estimate the cost of one particular iteration of BFGS as $12r^2 + 8r + (4r + cost_{CLL})count_{CLL} + cost_{grad}$; the cost of one CG and CGB iteration is $10r + (4r + cost_{CLL})count_{CLL} + cost_{grad}$. These estimates are obtained by inspecting the source code in (Nash, 1990).

## 5 Parameter learning

### 5.1 Logistic regression

Equation 2 can be rewritten as follows

$$P(X_0 = x_0|\mathbf{Z}) = \frac{e^{\mathbf{w}^{(x_0)^T}\mathbf{Z}}}{\sum_{x'_0=0}^{d_0-1} e^{\mathbf{w}^{(x'_0)^T}\mathbf{Z}}},$$

where we put $Z_0 = 1$, which corresponds to the intercept, and fix $\mathbf{w}^{(0)} = \mathbf{0}$. $\mathbf{w}^T$ denotes the transpose of $\mathbf{w}$. The conditional loglikelihood of the parameters given data is

$$CLL(\mathbf{w}) = \log \prod_{i=1}^{N} P(x_0^{(i)}|\mathbf{z}^{(i)})$$
$$= \sum_{i=1}^{N} (\mathbf{w}^{(x_0^{(i)})^T}\mathbf{z}^{(i)} - \log(\sum_{x'_0=0}^{d_0-1} e^{\mathbf{w}^{(x'_0)^T}\mathbf{z}^{(i)}})),$$

where $N$ is the number of observations in the data set. The gradient of the CLL is given by

$$\frac{\partial CLL(\mathbf{w})}{\partial w_k^{(x_0)}} =$$
$$\sum_{i=1}^{N} \left( 1_{\{x_0^{(i)}=x_0\}} z_k^{(i)} - \frac{e^{\mathbf{w}^{(x_0)^T}\mathbf{z}^{(i)}} z_k^{(i)}}{\sum_{x'_0=0}^{d_0-1} e^{\mathbf{w}^{(x'_0)^T}\mathbf{z}^{(i)}}} \right)$$

where $x_0 \in \mathcal{X}_0^-$. We note here that the data matrix $Z$ is a sparse matrix of indicators with 1s in

non-zero positions, thus the CLL and gradient functions can be implemented very efficiently. We estimate costs (in flops) according to above formulas: $cost_{CLL} = |Z|(d_0 - 1) + N(d_0 + 2), cost_{grad} = |Z|d_0 + 2Nd_0$, where $|Z|$ denotes number of 1s in the data matrix. Here we used the fact that the multiplication of two vectors ${\mathbf{w}^{(x_0)}}^T \mathbf{z}^{(i)}$ requires $|\mathbf{z}^{(i)}| - 1$ flops. In case of LR_Roos matrix $Z$ always contains exactly $1 + n - |pa(0)|$ 1s irrespective of graph complexity and dimension of attributes. For our mapping $|Z|$ depends on the sample. In the experiments we used the following heuristic to reduce $|Z|$: for each attribute $X_i$ we code the most frequent value as 0. We note that $|Z|$ is smaller in case of our mapping comparing to LR_Roos mapping, when the structure is not very complex (with regard to the number of parents and the domain size of the attributes) and becomes bigger for more complex structures.

### 5.2 Bayesian Network Classifiers

Here we follow the approach taken in (Greiner et al., 2005; Pernkopf and Bilmes, 2005). We write

$$\theta^j_{i|k} = P(x_j = i|\mathbf{x}_{pa(j)} = k).$$

We have constraints $\theta^j_{i|k} \geq 0$ and $\sum_{i=0}^{d_j-1} \theta^j_{i|k} = 1$. We reparameterize to incorporate the constraints on $\theta^j_{i|k}$ and use different parameters $\beta^j_{i|k}$ as follows

$$\theta^j_{i|k} = \frac{\exp \beta^j_{i|k}}{\sum_{l=0}^{d_j-1} \exp \beta^j_{l|k}}$$

The CLL is given by

$$CLL(\boldsymbol{\beta}) = \sum_{t=1}^{N} (\log P(\mathbf{x}^{(t)}) - \log \sum_{x_0^{(t)}=0}^{d_0-1} P(\mathbf{x}^{(t)}))$$

Further expansion may be obtained using the factorization of $P(\mathbf{x})$ and plugging in expressions for $\theta^j_{i|k}$. It is easy to see that

$$\frac{\partial \theta^j_{i'|k}}{\partial \beta^j_{i|k}} = \theta^j_{i'|k}(1_{\{i=i'\}} - \theta^j_{i|k}),$$

thus

$$\frac{\partial P(\mathbf{x})}{\partial \beta^j_{i|k}} = 1_{\{\mathbf{x}_{pa(j)}=k\}} P(\mathbf{x})(1_{\{x_j=i\}} - \theta^j_{i|k})$$

Simple calculations result in

$$\frac{\partial CLL(\beta)}{\partial \beta^j_{i|k}} = \sum_{t=1}^{N} \left( 1_{\{x_j^{(t)}=i, \mathbf{x}_{pa(j)}^{(t)}=k\}} - 1_{\{\mathbf{x}_{pa(j)}^{(t)}=k\}} \theta^j_{i|k} - \frac{\sum_{x_0^{(t)}=0}^{d_0-1} [P(\mathbf{x}^{(t)})(1_{\{x_j^{(t)}=i, \mathbf{x}_{pa(j)}^{(t)}=k\}} - 1_{\{\mathbf{x}_{pa(j)}^{(t)}=k\}} \theta^j_{i|k})]}{\sum_{x_0^{(t)}=0}^{d_0-1} P(\mathbf{x}^{(t)})} \right)$$

Note that for each $t$ and $j > 1$ only $d_0 d_j$ gradient values are to be considered. In order to obtain $\boldsymbol{\theta}$ from $\boldsymbol{\beta}$ we need $3|\boldsymbol{\beta}|$ flops, where $|\boldsymbol{\beta}|$ denotes the number of components of $\boldsymbol{\beta}$. From the formulas above we estimate $cost_{CLL} = 3|\boldsymbol{\beta}| + N(nd_0 + 2d_0 + 2)$ and $cost_{grad} = 3|\boldsymbol{\beta}| + N(2nd_0 + n + (2d_0 + 1)(3 + d_1 + \ldots + d_n))$.

Finally, we point out that the cost of the gradient is very close to the cost of the CLL in case of LR and is by a factor $2 + 2\overline{d}$ larger in case of BN. This suggests that CGB might be better than CG for BN, but it is very unlikely that it will be better also for LR and LR_Roos parameter spaces. Note that $cost_{CLL}$(BN) is very close to $cost_{CLL}$(LR_Roos), which strongly supports the fairness of our cost estimates.

### 5.3 Convergence issues

It is well known that LR has a concave loglikelihood function. Since BNCs whose canonical form is a perfect graph are equivalent to the LR models obtained by either mapping, it follows from the continuity of the mapping from $\boldsymbol{\theta}$ to $\mathbf{w}$, that the CLL function in the standard BN parameterization also has only global optima (Roos et al., 2005). This implies that all our algorithms should converge to the maximal CLL value.

It is important to pick good initial values for the parameters. The common approach to initialize BN parameters is to use the (generative) ML estimates. It is crucial for all three parameter spaces to avoid zero probabilities, therefore we use Laplace's rule, and add one to each frequency. After the initial values of $\boldsymbol{\theta}$ are obtained, we can derive starting values for $\mathbf{w}$ as well. We simply apply the mappings to obtain the initial values for the LR_Roos and LR parameters. This procedure guarantees that all algorithms have the same initial CLL value.

## 6 Experiments

For the experiments we used artificially generated data sets and data sets from the UCI Machine Learning Repository (Blake and Merz, 1998). Artificial data were generated from Bayesian Networks, where the parameter values were obtained by sampling each parameter from a Dirichlet distribution with $\boldsymbol{\alpha} = \mathbf{1}$. We generated data from both simple and complex structures in order to evaluate performance under different scenarios. We generated perfect graphs in the following way: for each attribute we select $i$ parents: the class node and $i-1$ previous nodes(whenever possible) according to indexing, where $i \in \{1, 2, 3\}$.

Table 1 lists the UCI data sets used in the experiments, and their properties. To discretize numeric variables, we used the algorithm described in (Fayyad and Irani, 1993). The ? values in the votes data set were treated as a separate value, not as a missing value.

Table 1: Data sets and their properties

| | #Samples | #Attr. | #Class | Max attr. dim. |
|---|---|---|---|---|
| Glass | 214 | 9 | 6 | 4 |
| Pima | 768 | 8 | 2 | 4 |
| Satimage | 4435 | 36 | 7 | 2 |
| Tic-tac-toe | 958 | 9 | 2 | 3 |
| Voting | 435 | 16 | 2 | 3 |

The fitted BNC structures for the UCI data were obtained by (1) connecting the class to all attributes, (2) using a greedy generative structure learning algorithm to add successive arcs. Step (2) was not applied for the Satimage data set for which we fitted the Naive Bayes model. All structures happened to be perfect graphs, thus there was no need for adjustment. In figure 2 we show the structure fitted to the Pima Indians data set. The other structures were of similar complexity.

Figure 3 depicts convergence curves for 9 optimization algorithms on the Pima Indians data.

Table 2 depicts statistics for different UCI and artificial data sets. For each data set $i$ we have starting CLL value $CLL^i_{ML}$ and maximal (over all algorithms) CLL value $CLL^i_{max}$. We define a threshold $t_i = CLL^i_{max} - 5\% * (CLL^i_{max} - CLL^i_{ML})$. For every algorithm we computed the number of iterations

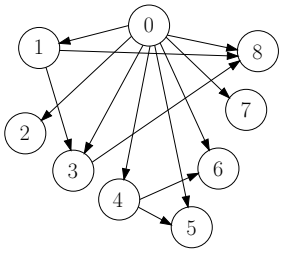



Figure 2: Structure fitted to Pima Indians data. The structure was constructed using a hill climber on the (penalized) generative likelihood.

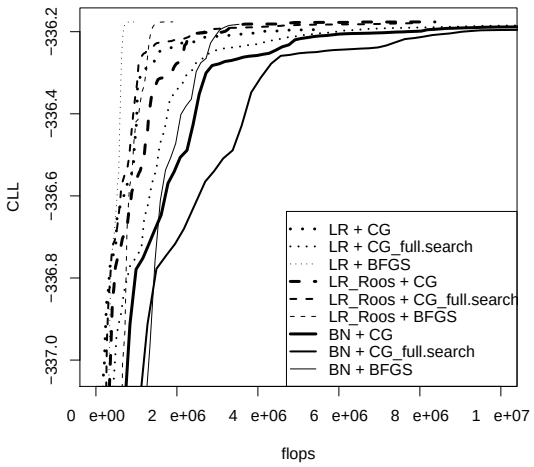



Figure 3: Optimization curves on the Pima Indians data.

(flops) needed to reach the threshold. For each data set these numbers are scaled dividing by the smallest value. The bound of $5\%$ was selected heuristically. This bound is big enough, so all algorithms were able to reach it before 200 iterations and before satisfying the stopping criterion. The bound is small enough, so algorithms make in general quite a few iterations before achieving it. There are some data sets, however, on which all algorithms converge fast to a very high CLL value, and a clear distinction in performance is visible using a very small bound. Pima Indians is an example (5% threshold is set to -337.037). We still see that table 2 contains a quite fair comparison, except that BFGS methods are underestimated for smaller bounds.

It is very interesting to notice that convergence

| Data Set | LR | | | LR_Roos | | | BN | | |
|---|---|---|---|---|---|---|---|---|---|
| | CG | CGB | BFGS | CG | CGB | BFGS | CG | CGB | BFGS |
| Glass | 2.1 | 1.9 | 1.0 | 3.0 | 2.0 | 1.1 | 2.4 | 1.9 | 1.0 |
| Pima | 1.3 | 1.5 | 1.8 | 1.5 | 1.2 | 1.8 | 1.0 | 1.0 | 1.8 |
| Satimage | 8.2 | 2.3 | 1.0 | 3.3 | 2.0 | 1.1 | 3.0 | 1.0 | 1.3 |
| Tic-tac-toe | 3.9 | 2.2 | 1.5 | 2.1 | 1.1 | 1.2 | 2.2 | 1.0 | 1.3 |
| Voting | 3.2 | 1.8 | 1.2 | 1.9 | 1.3 | 1.0 | 1.5 | 1.0 | 1.1 |
| 1.5.2-3.100 | 2.8 | 1.4 | 1.2 | 2.3 | 1.3 | 1.1 | 2.0 | 1.0 | 1.2 |
| 2.5.2-3.100 | 2.6 | 2.1 | 1.4 | 1.8 | 1.6 | 1.3 | 1.4 | 1.0 | 1.0 |
| 3.5.2-3.100 | 2.7 | 1.8 | 1.3 | 2.2 | 1.0 | 1.2 | 1.3 | 1.2 | 1.2 |
| 1.5.2-3.1000 | 2.2 | 2.2 | 2.2 | 1.5 | 1.5 | 2.5 | 1.0 | 1.0 | 1.5 |
| 2.5.2-3.1000 | 2.2 | 2.0 | 1.8 | 1.2 | 1.2 | 1.8 | 1.0 | 1.0 | 2.5 |
| 3.5.2-3.1000 | 3.6 | 2.2 | 1.0 | 1.8 | 1.6 | 1.0 | 1.3 | 1.3 | 1.0 |
| 1.30.2-3.1000 | 2.0 | 1.2 | 1.0 | 1.5 | 1.2 | 1.5 | 1.2 | 1.0 | 1.2 |
| 2.30.2-3.1000 | 2.0 | 1.8 | 2.2 | 1.6 | 1.2 | 1.4 | 1.4 | 1.0 | 1.2 |
| 3.30.2-3.1000 | 2.2 | 1.2 | 2.0 | 1.6 | 1.0 | 1.6 | 1.4 | 1.0 | 1.2 |
| 1.5.5.1000 | 3.1 | 3.2 | 1.8 | 2.3 | 2.3 | 1.5 | 1.1 | 1.1 | 1.0 |
| 2.5.5.1000 | 6.7 | 6.9 | 2.3 | 3.1 | 3.0 | 1.4 | 1.5 | 1.5 | 1.0 |
| 3.5.5.1000 | 4.8 | 4.2 | 1.8 | 2.3 | 2.2 | 1.1 | 1.7 | 1.7 | 1.0 |
| mean | 3.3 | 2.4 | 1.6 | 2.1 | 1.6 | 1.4 | 1.6 | 1.2 | 1.3 |

| Data Set | LR | | | LR_Roos | | | BN | | |
|---|---|---|---|---|---|---|---|---|---|
| | CG | CGB | BFGS | CG | CGB | BFGS | CG | CGB | BFGS |
| Glass | 1.0 | 2.4 | 3.3 | 2.6 | 4.1 | 8.3 | 5.9 | 8.1 | 12.6 |
| Pima | 1.0 | 2.4 | 1.6 | 1.8 | 2.5 | 3.1 | 4.0 | 6.1 | 11.0 |
| Satimage | 4.6 | 3.0 | 1.0 | 4.6 | 7.1 | 3.3 | 11.5 | 6.1 | 7.6 |
| Tic-tac-toe | 2.0 | 3.2 | 1.0 | 1.3 | 1.7 | 1.3 | 6.1 | 4.0 | 6.0 |
| Voting | 1.2 | 1.9 | 1.9 | 1.0 | 1.8 | 3.5 | 3.6 | 4.2 | 15.8 |
| 1.5.2-3.100 | 1.0 | 1.7 | 1.0 | 1.2 | 2.1 | 1.8 | 2.9 | 2.9 | 5.1 |
| 2.5.2-3.100 | 1.2 | 2.8 | 3.5 | 1.0 | 2.6 | 7.4 | 2.1 | 3.2 | 13.2 |
| 3.5.2-3.100 | 1.0 | 2.4 | 2.3 | 1.1 | 1.4 | 5.3 | 2.1 | 4.4 | 21.2 |
| 1.5.2-3.1000 | 1.0 | 2.0 | 1.7 | 1.0 | 1.9 | 3.9 | 1.5 | 2.4 | 5.5 |
| 2.5.2-3.1000 | 1.5 | 3.2 | 2.2 | 1.0 | 2.0 | 4.6 | 2.3 | 3.6 | 25.0 |
| 3.5.2-3.1000 | 1.7 | 2.6 | 1.4 | 1.0 | 2.2 | 3.5 | 2.2 | 4.3 | 13.5 |
| 1.30.2-3.1000 | 1.9 | 3.7 | 1.0 | 3.4 | 8.4 | 4.3 | 12.1 | 17.4 | 16.6 |
| 2.30.2-3.1000 | 1.0 | 3.1 | 2.2 | 1.2 | 3.2 | 3.3 | 4.9 | 7.4 | 11.8 |
| 3.30.2-3.1000 | 1.2 | 2.3 | 5.8 | 1.0 | 2.4 | 12.5 | 4.1 | 7.6 | 36.6 |
| 1.5.5.1000 | 1.1 | 2.8 | 1.5 | 1.0 | 2.4 | 1.9 | 1.6 | 2.4 | 2.8 |
| 2.5.5.1000 | 2.3 | 5.6 | 11.1 | 1.0 | 2.2 | 10.1 | 1.7 | 2.4 | 11.6 |
| 3.5.5.1000 | 2.6 | 5.4 | 101.5 | 1.0 | 2.3 | 98.1 | 2.4 | 3.8 | 137.5 |
| mean | 1.6 | 3.0 | 8.5 | 1.5 | 3.0 | 10.4 | 4.2 | 5.3 | 20.8 |

Table 2: Convergence speed in terms of iterations (top) and flops (bottom). Artificial data sets are named in the following way: [number of parents of each attribute].[number of attributes].[domain size of each attribute].[sample size]. Domain size denoted as 2-3 is randomly chosen between 2 and 3 with equal probabilities.

with regard to iterations is faster in BN and LR_Roos parameter spaces. It seems that overparameterization is advantageous in this respect. This might be explained by the fact that there is only a single global optimum in the LR case, whereas there are many global optima in case of BN and LR_Roos. Thus, in the latter case one can 'quickly' converge to the closest optimum. Overparameterization, however, is also an additional burden, as witnessed by the flops count; this is especially true for the BFGS method.

We observe that BFGS is generally winning, with CGB being close, in terms of iterations. Considering flops, CGB is definitely losing to CG in all parameterizations. CG seems to be the best technique, though it might be very advantageous to use BFGS for simple (with respect to the number of parents and the domain size of attributes) structures, in combination with our LR mapping.

We have both theoretical and practical evidence that our LR parameterization is the best for relatively simple structures. So the general conclusion is to use LR + BFGS, LR + CG and LR_Roos + CG in order of growing structure complexity. The size of the domain and the number of parents mainly influence our choice. On the basis of flop counts, the BN parameterization is never preferred in our experiments. Finally, we note that our LR parameterization was the best on 4 out of 5 UCI data sets.

## 7 Conclusion

We have studied the efficiency of discriminative scoring of BNCs using alternative parameterizations of the conditional distribution of the class variable. In case the canonical form of the BNC is a perfect graph, there is a choice between at least three parameterizations. We found out that it is wise to exploit perfectness by optimizing in LR or LR_Roos spaces. Based on the experiments we have performed, we would suggest to use LR + BFGS, LR + CG and LR_Roos + CG in order of growing structure complexity. If only one method is to be selected, we suggest LR_Roos + CG. It works pretty well on any type of data set, plus the mapping is very straightforward, which makes the initialization step easy to implement.

## References


J. A. Anderson. 1982. Logistic discrimination. In P. R. Krishnaiah and L. N. Kanal, editors, *Classification, Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics*, pages 169–191. North-Holland.

C.L. Blake and C.J. Merz. 1998. UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/mlrepository.html].

U. Fayyad and K. Irani. 1993. Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of IJCAI-93 (volume 2)*, pages 1022–1027. Morgan Kaufmann.

R. Greiner, S. Xiaoyuan, B. Shen, and W. Zhou. 2005. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59:297–322.

T. Minka. 2001. Algorithms for maximum-likelihood logistic regression. Technical Report Statistics 758, Carnegie Mellon University.

J. C. Nash. 1990. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation (2nd ed.)*. Hilger.

F. Pernkopf and J. Bilmes. 2005. Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 657–664, New York. ACM Press.

W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. 1992. *Numerical Recipes in C: the art of scientific computing*. Cambridge.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. 2005. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59:267–296.

G. Santafé, J. A. Lozano, and P. Larrañaga. 2005. Discriminative learning of Bayesian network classifiers via the TM algorithm. In L. Godo, editor, *ECSQARU 2005*, volume 3571 of *Lecture Notes in Computer Science*, pages 148–160. Springer.

C. Sutton and A. McCallum. 2006. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press. To appear.

W.N. Venables and B.D. Ripley. 2002. *Modern Applied Statistics with S (fourth edition)*. Springer, New York.

# The *Independency tree* model: a new approach for clustering and factorisation

M. Julia Flores and José A. Gámez
Computing Systems Department / SIMD ($i^3\mathcal{A}$)
University of Castilla-La Mancha
Albacete, 02071, Spain

Serafín Moral
Departamento de Ciencias de la Computación e I. A.
Universidad de Granada
Granada, 18071, Spain

**Abstract**

Taking as an inspiration the so-called *Explanation Tree* for abductive inference in Bayesian networks, we have developed a new clustering approach. It is based on exploiting the variable independencies with the aim of building a tree structure such that in each leaf all the variables are independent. In this work we produce a structure called *Independency tree*. This structure can be seen as an extended probability tree, introducing a new and very important element: a list of probabilistic single potentials associated to every node. In the paper we will show that the model can be used to approximate a joint probability distribution and, at the same time, as a hierarchical clustering procedure. The Independency tree can be learned from data and it allows a fast computation of conditional probabilities.

## 1 Introduction

In the last years the relevance of unsupervised classification within data mining processing has been remarkable. When dealing with large number of cases in real applications, the identification of common features that allows the formation of groups/clusters of cases seems to be a powerful capability that both simplifies the data processing and also allows the user to understand better the trend(s) followed in the registered cases.

In the data mining community this *descriptive* task is known as *cluster analysis* (Anderberg, 1973; Duda et al., 2001; Kaufman and Rousseeuw, 1990; Jain et al., 1999), that is, getting a decomposition or partition of a data set into groups in such a way that the objects in one group are similar to each other but as different as possible from the objects in other groups. In fact, as pointed out in (Hand et al., 2001, pg. 293), we could distinguish two different objectives in this descriptive task: (a) *segmentation*, in which the aim is simply to partition the data in a *convenient* way, probably using only a small number of the available variables; and (b) *decomposition*, in which the aim is to see whether the data is composed (or not) of natural subclasses, i.e., to discover whether the overall population is heterogeneous. Strictly speaking cluster analysis is devoted to the second goal although, in general, the term is widely used to describe both segmentation and cluster analysis problems.

In this work we only deal with *categorical* or *discrete* variables and we are closer to segmentation than (strictly speaking) to cluster analysis. Our proposal is a method that in our opinion has several good properties: (1) it produces a tree-like graphical structure that allows us to visually describe each cluster by means of a configuration of the relevant variables for

that segment of the population; (2) it takes advantage of the identification of contextual independencies in order to build a decomposition of the joint probability distribution; (3) it stores a joint probability distribution about all the variables, in a simple way, allowing an efficient computation of conditional probabilities. An example of an independence tree is given in figure 1, which will be thoroughly described in the following sections.

The paper is structured as follows: first, in Section 2 we give some preliminaries in relation with our proposed method. Section 3 describes the kind of model we aim to look for, while in Section 4 we propose the algorithm we have designed to discover it from data. Section 5 describes the experiments carried out. Finally, Section 6 is devoted to the conclusions and to describe some possible lines in order to continue our research.

## 2 Preliminaries

Different types of clustering algorithms can be found in the literature differing in the type of approach they follow. Probably the three main approaches are: partition-based clustering, hierarchical clustering, and probabilistic model-based clustering. From them, the first two approaches yield a *hard* clustering in the sense that clusters are exclusive, while the third one yields a *soft* clustering, that is, an object can belong to more than one cluster following a probability distribution. Because our approach is somewhat related to hierarchical and probabilistic model-based clustering we briefly comment on these types of clustering.

Hierarchical clustering (Sneath and Sokal, 1973) returns a tree-like structure called dendrogram. This structure reflects the way in which the objects in the data set have been merged from single points to the whole set or split from the whole set to single points. Thus, there are two distinct types of hierarchical methods: *agglomerative* (by merging two clusters) and *divisive* (by splitting a cluster). Although our method does not exactly belong to hierarchical clustering, it is closer to hierarchical divisive methods than to any other clustering approach (known by us). In concrete, it works as *monothetic* divisive clustering algorithms (Kaufman and Rousseeuw, 1990), that split clusters using one variable at a time, but differs from classical divisive approaches in the use of a Bayesian score to decide which variable is chosen at each step, and because branches are not completely developed.

With respect to probabilistic clustering, although our method produces a hard clustering, it is somehow related to it because probability distributions are used to complete the information about the discovered model. Probabilistic model-based clustering is usually modelled as a mixture of models (see e.g. (Duda et al., 2001)). Thus, a hidden random variable is added to the original observed variables and its states correspond with the components of the mixture (the number of clusters). In this way we move to a problem of learning from unlabelled data and usually EM algorithm (Dempster et al., 1977) is used to carry out the learning task when the graphical structure is fixed and *structural* EM (Friedman, 1998) when the graphical structure has also to be discovered (Peña et al., 2000). Iterative approaches have been described in the literature (Cheeseman and Stutz, 1996) in order to discover also the number of clusters (components of the mixture). Notice that although this is not the goal of the learning task, the structures discovered can be used for approximate inference (Lowd and Domingos, 2005), having the advantage over general Bayesian networks (Jensen, 2001) that the learned graphical structure is, in general, simple (i.e. naive Bayes) and so inference is extremely fast.

## 3 *Independency tree* model

If we have the set of variables $\mathbf{X} = \{X_1, \ldots, X_m\}$ regarding a certain domain, an independent probability tree for this set of variables is a tree such that (e.g. figure 1):

- Each inner node $N$ is labelled by a variable $\mathsf{Var}(N)$, and this node has a child for each one of the possible values (states) of $\mathsf{Var}(N)$.

When a variable is represented by a node in the tree it implies that this variable partitions the space from now on (from this point to the farther branches in the tree) depending on its (nominal) value. So, a variable cannot appear again (deeper) in the tree.

- Each node $N$ will have an associated list of potentials, $\mathsf{List}(N)$, with each of the potentials in the list storing a marginal probability distribution for one of the variables in $\mathbf{X}$, including always a potential for the variable $\mathsf{Var}(N)$.

  This list appears in Figure 1 framed into a dashed box.

- For any path from the root to a leaf, each one of the variables in $\mathbf{X}$ appears uniquely and exactly once in the the lists associated to the nodes along the path. This potential will determine the conditional probability of the variables given the values of the variables on the path from the root to the list containing the potential.

A configuration is a subset of variables $\mathbf{Y} \subseteq \{X_1, \ldots, X_m\}$ together with a concrete value $Y_j = y_j$ for each one of the variables $Y_j \in \mathbf{Y}$. Each node $N$ has an associated configuration determined for the variables in the path from the root to node $N$ (excluding $\mathsf{Var}(N)$) with the values corresponding to the children we have to follow to reach $N$. This configuration will be denoted as $\mathsf{Conf}(N)$.

An independent probability tree represents a joint probability distribution, $p$, about the variables in $\mathbf{X}$. If $\mathbf{x} = (x_1, \ldots, x_m)$, then

$$p(\mathbf{x}) = \prod_{i=1}^{m} \mathcal{P}_{\mathbf{x}(i)}(x_i)$$

where $\mathcal{P}_{x(i)}$ is the potential for variable $X_i$ which is in the path from the root to a leaf determined by configuration $\mathbf{x}$ (following in each inner node $N$ with variable $\mathsf{Var}(N) = X_j$, the child corresponding to the value of $X_j \in \mathbf{x}$).

This decomposition is based on a set of independencies among variables $\{X_1, \ldots, X_m\}$. Assume that $\mathsf{VL}(N)$ is the set of variables of the potentials in $\mathsf{List}(N)$, and that $\mathsf{DVL}(N)$[1] is the union of sets $\mathsf{VL}(N')$, where $N'$ is a node in a path from $N$ to a leaf, then the independencies are generated from the following statement: *Each variable $X$ in $\mathsf{VL}(N)$ - $\mathsf{Var}(N)$ is independent of the variables $(\mathsf{VL}(N) - \{X\}) \cup \mathsf{DVL}(N)$ given the configuration $\mathsf{Conf}(N)$*; i.e. each variable in the list of a node is independent of the other variables in that list and of the variables in its descendants, given the configuration associated to the node.

An independent probability tree also defines a partition of the set of possible values of variables in $\mathbf{X}$. The number of clusters is the number of leaves. If $N$ is a leaf with associated configuration $\mathsf{Conf}(N)$, then this group is given by all set of values $\mathbf{x}$ that are compatible with $\mathsf{Conf}(N)$ (they have the same value for all the variables in $\mathsf{Conf}(N)$). For example, in figure 1 the configuration $\mathbf{X} = \{0, 1, 0, 1, 0\}$ would fall on the second leaf since $X_1$=0 and $X_2$=1. It is assumed that the probability distribution of the other variables in the cluster are given by the potentials in the path defined by the configuration, for example, P($X_3$=0)=1.

In this way, with an independence tree we are able of accomplishing two goals in one: (1)The variables are partitioned in a hierarchical way that gives us at the same time a clustering result; (2)The probability value of every configuration of the variables.

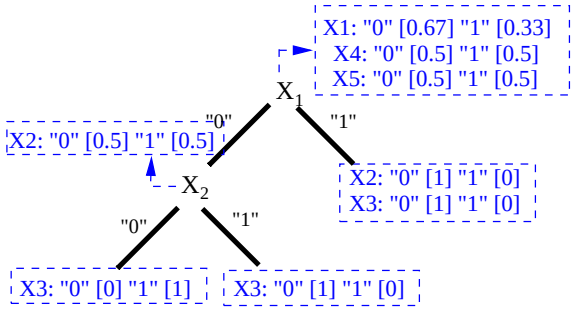



Figure 1: Illustrative independence tree structure learned from the exclusive dataset.

[1] D stands for "Descendants".

Our structure seeks to keep in leaves those variables that remain independent. At the same time, if the distribution of one variable is shared by several leaves, we try to store it in their common ascendant, to avoid repeating it in all the leaves. For that reason, when one variable appears in a list for a node $N_j$ it means that this distribution is common for all levels from here to a leaf. For example, in figure 1, the binary variable $X_4$ has a uniform (1/2,1/2) distribution for all leaf cases (also for intermediate ones), since it is associated to the root node. On the other hand, we can see how $X_2$ distribution varies depending on the branch (left or right) we take from the root, that is, when $X_1 = 0$ the behaviour of variable $X_2$ is uniform whereas being 1 the value of $X_1$, $X_2$ is determined to be 0.

The intuition underlying this model is based on the idea that inside each cluster the variables are independent. When we have a set of data, groups are defined by common values in certain variables, having the other variables random variations. Imagine that we have a database with characteristics of different animals including mammals and birds. The presence of these two groups is based on the existence of dependencies between the variables (two legs is related with having wings and feathers). Once these variables are fixed, there can be another variables (size, colour) that can have random variations inside each group, but that they do not define new subcategories.

Of course, there are some other possible alternatives to define clustering. This is based on the idea that when all the variables are independent, then to subdivide the population is not useful, as we have a simple method to describe the joint behaviour of the variables. However, if some of the variables are dependent, then the values of some basic variables could help to determine the values of the other variables, and then it can be useful to divide the population in groups according to the values of these basic variables.

Another way of seeing it is that having independent variables is the simplest model we can have. So, we determine a set of categories such that each one is described in a very simple way (independent variables). This is exploited by another clustering algorithms as EM-based AutoClass clustering algorithm (Cheeseman and Stutz, 1996).

An important fact of this model is that it is a generalisation of some usual models in classification, the Naive Bayes model (a tree with only one inner node associated to the class and in its children all the rest of variables are independent), and the classification tree (a tree in which the list of each inner node only contains one potential and the potential associated to the class variable always appears in the leaves). This generalization means that our model is able of representing the same conditional probability distribution of the class variable with respect to the rest of the variables, taking as basis the same set of associated independencies.

From this model one may want to apply two kinds of operations:

• The production of clusters and their characterisation. A simple in-depth from root until every leaf node access of the tree will give us the corresponding clusters, and also the potential lists associated to the path nodes will indicate the behaviour of the other variables not in the path.

• The computation of the probability for a certain complete configuration with a value for each variable: $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$. This algorithm is recursive and works as follows:

getProb(Configuration $\{x_1, x_2, \ldots, x_m\}$,Node $N$)
  $Prob \leftarrow 1$
  1 For all Potential $\mathcal{P}_j \in$ List(N)
    1.1. $X_j \leftarrow$ Var($\mathcal{P}_j$)
    1.2. $Prob \leftarrow Prob \cdot \mathcal{P}_j(X_j = x_j)$
  2 If $N$ is a leaf then return $Prob$
  3 Else
    3.1. $X_N \leftarrow$ Var(N)
    3.2. Next Node $N' \leftarrow$ Branch_child($X_N = x_N$)
    3.3. return $Prob$·getProb($\{x_1, x_2, \ldots, x_m\}$,$N'$)

If we have a certain configuration we should go through the tree from root until leaves taking the corresponding branches. Every time we reach a node, we have to use the single potentials in the associated list to multiply the value of probability by the values of the potentials corresponding to this configuration.

It is also possible to compute the probability for an interest variable $Z$ conditioned to

a generic configuration (a set of observations): $\mathbf{Y} = \mathbf{y}$. This can be done in two steps: 1) Transform the independent probability tree into a probability tree (Salmerón et al., 2000); 2) Make a marginalisation of the probability tree by adding in all the variables except in $Z$ as describe in (Salmerón et al., 2000).

In the following we describe the first step. It is a recursive procedure that visits all nodes from root to leaves. Each node can pass to its children a float, $Prob$ (1 in the root) and a potential $\mathcal{P}$ which depends of variable $Z$ (empty in the root). Each time a node is visited, the following operations are carried out:

• All the potentials in $\mathsf{List}(N)$ are examined and removed. For any potential, depending on its variable $X_j$ we proceed as follows:

- If $X_j \neq Z$, then if $X_j = Y_k$ is in the observations configuration, $Prob$ is multiplied by the value of this potential for $Y_k = y_k$.

- If $X_j \neq Z$ and $X_j$ does not appear in the observations configuration, then the potential is ignored.

- If $X_j = Z$ and $X_j$ is the one associated to node $N$, then we transform each one of the children of $Z$, but multiplying $Prob$ by the value of potential in $Z = z$, before transforming the child corresponding to this value.

- If $X_j = Z$ and $X_j$ is not the one associated to node $N$, then the potential of $X_j$ is stored in $\mathcal{P}$.

• After examining the list of potentials, we proceed:

- If $N$ is not a leaf node, then we transform its children.

- If $N$ is a leaf node and $\mathcal{P} = \emptyset$, we assign the value $Prob$ to this node.

- If $N$ is a leaf node and $\mathcal{P} \neq \emptyset$, we make $N$ an inner node with variable $Z$: For each value $Z = z$, build a node $N'_z$ which is a leaf node with a value equal to $Prob \odot \mathcal{P}(z)$. Make $N'_z$ a child of $N$.

This procedure is fast: it is linear in the the size of the independent probability tree (considering the number of values of $Z$ constant). The marginalisation in the second step has the same time complexity. In fact, this marginalization can be done by adding for each value $Z = z$ the values of the leaves that are compatible with this value (compatibility means that to follow this path we do not have to assume $Z = z'$ with $z \neq z'$), which is linear too.

## 4 Our clustering algorithm

In this section we are going to describe how an independent probability tree can be learned from a database $D$, with values for all the variables in $\mathbf{X} = \{X_1, \ldots, X_m\}$.

The basics of the algorithm are simple. It tries to determine for each node, the variable with a strongest degree of dependence with the rest of remaining variables. This variable will be assigned to this node, repeating the process with its children until all the variables are independent.

For this, we need a measure of the degree of dependence of two variables $X_i$ and $X_j$ in database $D$. The measure should be centered around 0, in such a way that the variables are considered dependent if and only if the measure is greater than 0. In this paper, we consider the K2 score (Cooper and Herskovits, 1992), measuring the degree of dependence as the difference in the logarithm of the K2 score of $X_j$ conditioned to $X_i$ minus the logarithm of K2 score of marginal $X_j$, i.e. the difference between the logarithms of the K2 scores of two networks with two variables: one in which $X_i$ is a parent of $X_j$ and other in which the two variables are not connected. Let us call this degree of dependence $\mathsf{Dep}(X_i, X_j|D)$. This is a non-symmetrical measure and should be read as the influence of $X_i$ on $X_j$, however in practice the differences between $\mathsf{Dep}(X_i, X_j|D)$ and $\mathsf{Dep}(X_j, X_i|D)$ are not important.

In any moment, given a variable $X_i$ and a database $D$, we can estimate a potential $\mathcal{P}_i(D)$ for this variable in the database. This potential is the estimation of the marginal probability of $X_i$. Here we assume that this is done by counting the absolute frequencies of each one of the values in the database.

The algorithm starts with a list of variables $L$ which is initially equal to $\{X_1, \ldots, X_m\}$ and a database equal to the original one $D$, then it

determines the root node $N$ and its children in a recursive way. For that, for any variable $X_i$ in the list, it computes

$$\mathsf{Dep}(X_i|D) = \sum_{X_j \in L} \mathsf{Dep}(X_i, X_j|D)$$

Then, the variable $X_k$ with maximum value of $\mathsf{Dep}(X_i|D)$ is considered.

If $\mathsf{Dep}(X_k|D) > 0$, then we assign variable $X_k$ to node $N$ and we add potential $\mathcal{P}_k(D)$ to the list $\mathsf{List}(N)$, removing $X_k$ for $L$. For all the remaining variables $X_i$ in $L$, we compute $\mathsf{Dep}(X_i, X_j|D)$ for $X_j \in L(j \neq i)$, and $X_j = X_k$. If all these values are less or equal than 0, then we add potential $\mathcal{P}_i(D)$ to $\mathsf{List}(N)$ and remove $X_i$ from $L$; i.e. we keep in this node the variables which are independent of the rest of variables, including the variable in the node. Finally, we build a child of $N$ for each one of the values $X_k = x_k$. This is done by calling recursively to the same procedure, but with the new list of nodes, and changing database $D$ to $D[X_k = x_k]$, where $D[X_k = x_k]$ is the subset of $D$ given by those cases in which variable $X_k$ takes the value $x_k$.

If $\mathsf{Dep}(X_k|D) \leq 0$, then the process is stopped and this is a leaf node. We build $\mathsf{List}(N)$, by adding potential $\mathcal{P}_i(D)$ for any variable $X_i$ in $L$.

In this algorithm, the complexity of each node computation is limited by $O(m^2.n)$ where $m$ is the number of variables and $n$ is the database size. The number of leaves is limited by the database size $n$ multiplied by the maximum number of cases of a variable, as a leaf with only one case of the database is never branched. But usually the number of nodes is much lower.

In the algorithm, we make some approximations with respect to the independencies represented by the model. First, we only look at one-to-one dependencies, and not to joint dependencies. It can be the case that $X_i$ is independent of $X_j$ and $X_k$ and it is not independent of $(X_j, X_k)$. However, testing these independents is more costly and we do not have the possibility of a direct representation of this in the model. This assumption is made by other Bayesian networks learning algorithms such as PC (Spirtes et al., 1993), where a link between two nodes is deleted if these nodes are marginally independent.

Another approximation is that it is assumed that all the variables are independent in a leaf when $\mathsf{Dep}(X_k|D) \leq 0$, even if some of the terms we are adding are positive. We have found that this is a good compromise criterion to limit the complexity of learned models.

## 5 Experiments

To make an initial evaluation of the *Independence Tree* (IndepT) model, we decided to compare it with other well known unsupervised classification techniques that are also based on Probabilistic Graphical Models: learning of a Bayesian network (by a standard algorithm as PC) and also Expectation-Maximisation with a Naive Bayes structure, which uses cross-validation to decide the number of clusters. Because we produce a probabilistic description of the dataset, we use the log-likelihood (logL) of the data given the model to score a given clustering. By using the logL as goodness measure we can compare our approach with other algorithms for probabilistic model-based unsupervised learning: probabilistic model-based clustering and Bayesian networks. This is a direct evaluation of the procedures as methods to encode a complex joint probability distribution. At the same time, it is also an evaluation of our method from the clustering point of view, showing whether the proposed segmentation is useful for a simple description of the population.

Then, the basic steps we have followed for the three procedures [IndepT,PC-Learn,EM-Naive] are:

1. Divide the data cases into a training or data set ($S_D$) and a test set ($S_T$), using (2/3,1/3).
2. Build the corresponding model for the cases in $S_D$.
3. Compute the log-likelyhood of the obtained model over the data in $S_T$.

Among the tested cases, apart from easy synthetical data bases created to check the expected and right clusters, we have looked for real sets of cases. Some of them have been taken

from the UCI datasets repository (Newman et al., 1998) and others from real applications related to our current research environment.

We will indicate the main remarks about all the evaluated cases:

• Case 1: *exclusive*: This is a very simple dataset with five binary variables, from $X_1$ to $X_5$. The three first variables have an *exclusive* behaviour, that is, if one of them is 1 the other two will be 0. On the other hand, $X_4$ and $X_5$ are independent with the three first ones and also with each other. This example is interesting not especially for the LogL comparison, but mainly to see the interpretation of the tree given in figure 1.

• Case 2: *tic-tac-toe*: Taken from UCI repository and it encodes the complete set of possible board configurations at the end of tic-tac-toe games. It presents 958 cases and 9 attributes (one per each game square).

• Cases 3: *greenhouses*: Data cases taken from real greenhouses located in Almería (Spain). There are four distinct sets of cases, here we indicate them with denotation (case)={num_cases, num_attributes}: ($3A$)= {1240,8}, ($3B$)= {1465,17}, ($3C$)= {1318,33}, ($3D$)= {1465,6}.

• Cases 4: *sheep*: Datasets taken from the work developed in (Flores and Gámez, 2005) where the historical data of the different animals were registered and used to analyse their genetic merit for milk production directed to Manchego cheese. There are two datasets with 3087 cases, $4A$ with 24 variables/attributes and $4B$, where the attribute `breeding value` (that could be interpreted as class attribute) has been removed.

• Case 5: *connect4*: Also downloaded from the UCI repository and it contains all legal 8-play positions in the game of connect-4 in which neither player has won yet, and in which the next move is not forced. There are 67557 cases and 42 attributes, each corresponding to one connect-4 square[2].

The results of the experiments can be found in figure 1 and in table 1. Figure 1 presents the tree learned in the *exclusive* case. As it can be observed, the tree really captures what is happening in the problem: $X_4$ and $X_5$ are independent and placed in the list of root node and then one of the other variables is looked up, if its value is 1, then the values of the other two variables is completely determined and we stop. If its value is 0, then another variable has to be examined.

| case | IndepT | PC-Learn | EM-Naive |
|---|---|---|---|
| 1 | -61.33 | -62.88 | -119.67 |
| 2 | -1073.94 | -2947.85 | -3535.11 |
| $3A$ | -3016.92 | -2644.38 | -4297.21 |
| $3B$ | -2100.40 | -3584.06 | -6266.71 |
| $3C$ | -4956.04 | -7630.98 | -15145.15 |
| $3D$ | -1340.04 | -2770.15 | -4223.45 |
| $4A$ | -6853.12 | -18074.69 | -32213.29 |
| $4B$ | -6802.94 | -17357.80 | -31244.28 |
| 5: | -465790.73 | -349631.35 | -794415.07 |

Table 1: Comparison in terms of the log-likelyhood value for all cases (datasets).

With respect to the capacity of approximating the joint probability distribution, we can see in table 1 that our method always provides greater values of logL than EM-Naive in all the datasets. With respect to PC algorithm, the independent tree wins in all the situations except in two of them (cases $3A$ and 5). This is a remarkable result as our model has some limitations in representing sets of independencies that can be easily represented by a Bayesian network (for example a Markov chain), however its behaviour is usually better. We think that this is due to two main aspects: it can represent asymmetrical independencies and it is a simple model (which is always a virtue).

## 6 Concluding remarks and future work

In this paper we have proposed a new model for representing a joint probability distribution in a compact way, which can be used for fast computation of conditional probability distributions. This model is based on a partition of the state space, in such a way that in each group all the

[2]Actually, only the half of the cases have been used.

variables are independent. In this sense, it is at the same time a clustering algorithm.

In the experiments with real and synthetical data we have shown its good behaviour as a method of approximating a joint probability, providing results that are better (except for two cases for the PC algorithm) to the factorisation provided by an standard Bayesian networks learning algorithm (PC) and by EM clustering algorithm. In any case, more extensive experiments are necessary in order to compare it with another clustering and factorisation procedures.

We think that this model can be improved and exploited in several ways: (1) Some of the clusters can have very few cases or can be very similar in the distributions of the variables. We could devise a procedure to joint these clusters; (2) we feel that the different number of values of the variables can have some undesirable effects. This problem could be solved by considering binary trees in which the branching is determined by a partition of the set of possible values of a variable in two parts. This could allow to extend the model to continuous variables; (3) we could determine different stepping branching rules depending of the final objective: to approximate a joint probability or to provide a simple (though not necessarily exhaustive) partition of the state space that could help us to have an idea of how the variables take their values; (4) to use this model in classification problems.

**Acknowledgments**

This work has been supported by Spanish MEC under project TIN2004-06204-C03-{02,03}.

## References

M.R. Anderberg. 1973. *Cluster Analysis for Applications*. Academic Press.

P. Cheeseman and J. Stutz. 1996. Bayesian classification (AUTOCLASS): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press/MIT Press.

G.F. Cooper and E.A. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

A. P. Dempster, N. M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1:1–38.

R.O. Duda, P.E. Hart, and D.J. Stork. 2001. *Pattern Recognition*. Wiley.

M.J. Flores and J. A. Gámez. 2005. Breeding value classification in manchego sheep: A study of attribute selection and construction. In *Kowledge-Based Intelligent Information and Enginnering Systems. LNAI*, volume 3682, pages 1338–1346. Springer Verlag.

N. Friedman. 1998. The Bayesian structural EM algorithm. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 129–138. Morgan Kaufmann.

D. Hand, H. Mannila, and P. Smyth. 2001. *Principles of Data Mining*. MIT Press.

A.K. Jain, M.M. Murty, and P.J. Flynn. 1999. Data clustering: a review. *ACM Computing Surveys*, 31:264–323.

F. V. Jensen. 2001. *Bayesian Networks and Decision Graphs*. Springer Verlag.

L. Kaufman and P. Rousseeuw. 1990. *Finding Groups in Data*. John Wiley and Sons.

D. Lowd and P. Domingos. 2005. Naive Bayes models for probability estimation. In *Proceedings of the 22nd ICML conference*, pages 529–536. ACM Press.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. 1998. UCI repository of machine learning databases.

J.M. Peña, J.A. Lozano, and P. Larrañaga. 2000. An improved Bayesian structural EM algorithm for learning Bayesian networks for clustering. *Pattern Recognition Letters*, 21:779–786.

A. Salmerón, A. Cano, and S. Moral. 2000. Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413.

P.H. Sneath and R.R. Sokal. 1973. *Numerical Taxonomy*. Freeman.

P. Spirtes, C. Glymour, and R. Scheines. 1993. *Causation, Prediction and Search*. Springer Verlag.

# Learning the Tree Augmented Naive Bayes Classifier from incomplete datasets

Olivier C.H. François and Philippe Leray
LITIS Lab., INSA de Rouen, BP 08, av. de l'Université
76801 Saint-Etienne-Du-Rouvray, France.

**Abstract**

The Bayesian network formalism is becoming increasingly popular in many areas such as decision aid or diagnosis, in particular thanks to its inference capabilities, even when data are incomplete. For classification tasks, Naive Bayes and Augmented Naive Bayes classifiers have shown excellent performances. Learning a Naive Bayes classifier from incomplete datasets is not difficult as only parameter learning has to be performed. But there are not many methods to efficiently learn Tree Augmented Naive Bayes classifiers from incomplete datasets. In this paper, we take up the structural EM algorithm principle introduced by (Friedman, 1997) to propose an algorithm to answer this question.

## 1 Introduction

Bayesian networks are a formalism for probabilistic reasoning increasingly used in decision aid, diagnosis and complex systems control. Let $\mathbb{X} = \{X_1, \ldots, X_n\}$ be a set of discrete random variables. A Bayesian network $\mathcal{B} =< \mathcal{G}, \Theta >$ is defined by a directed acyclic graph $\mathcal{G} =< \mathbb{N}, \mathbb{U} >$ where $\mathbb{N}$ represents the set of nodes (one node for each variable) and $\mathbb{U}$ the set of edges, and parameters $\Theta = \{\theta_{ijk}\}_{1 \leqslant i \leqslant n, 1 \leqslant j \leqslant q_i, 1 \leqslant k \leqslant r_i}$the set of conditional probability tables of each node $X_i$ knowing its parents' state $Pi$ (with $r_i$ and $q_i$ as respective cardinalities of $X_i$ and $P_i$).

If $\mathcal{G}$ and $\Theta$ are known, many inference algorithms can be used to compute the probability of any variable that has not been measured conditionally to the values of measured variables. Bayesian networks are therefore a tool of choice for reasoning in uncertainty, based on incomplete data, which is often the case in real applications.

It is possible to use this formalism for classification tasks. For instance, the Naive Bayes classifier has shown excellent performance. This model is simple and only need parameter learning that can be performed with incomplete datasets. Augmented Naive Bayes classifiers (with trees, forests or Bayesian networks), which often give better performances than the Naive Bayes classifier, require structure learning. Only a few methods of structural learning deal with incomplete data.

We introduce in this paper a method to learn Tree Augmented Naive Bayes (TAN) classifiers based on the expectation-maximization (EM) principle. Some previous work by (Cohen et al., 2004) also deals with TAN classifiers and EM principle for partially unlabeled data. In there work, only the variable corresponding to the class can be partially missing whereas any variable can be partially missing in the approach we propose here.

We will therefore first recall the issues relating to structural learning, and review the various ways of dealing with incomplete data, primarily for parameter estimation, and also for structure determination. We will then examine the structural EM algorithm principle, before proposing and testing a few ideas for improvement based on the extension of the Maximum Weight Spanning Tree algorithm to deal with incomplete data. Then, we will show how to use the introduced method to learn the well-known Tree Augmented Naive Bayes classifier from incomplete datasets and we will give some experiments on real data.

## 2 Preliminary remarks

### 2.1 Structural learning

Because of the super-exponential size of the search space, exhaustive search for the best structure is impossible. Many heuristic methods have been proposed to determine the structure of a Bayesian network. Some of them rely on human expert knowledge, others use real data which need to be, most of the time, completely observed.

Here, we are more specifically interested in score-based methods. Primarily, greedy search algorithm adapted by (Chickering et al., 1995) and maximum weight spanning tree (MWST) proposed by (Chow and Liu, 1968) and applied to Bayesian networks in (Heckerman et al., 1995). The greedy search carried out in directed acyclic graph (DAG) space where the interest of each structure located near the current structure is assessed by means of a BIC/MDL type measurement (Eqn.1)[1] or a Bayesian score like BDe (Heckerman et al., 1995).

$$BIC(\mathcal{G},\Theta) = \log P(\mathcal{D}|\mathcal{G},\Theta) - \frac{\log N}{2}\mathrm{Dim}(\mathcal{G}) \quad (1)$$

where $\mathrm{Dim}(\mathcal{G})$ is the number of parameters used for the Bayesian network representation and $N$ is the size of the dataset $\mathcal{D}$.

The BIC score is decomposable. It can be written as the sum of the local score computed for each node as $BIC(\mathcal{G},\Theta) = \sum_i bic(X_i, P_i, \Theta_{X_i|P_i})$ where $bic(X_i, P_i, \Theta_{X_i|P_i}) =$

$$\sum_{X_i=x_k}\sum_{P_i=pa_j} N_{ijk}\log\theta_{ijk} - \frac{\log N}{2}\mathrm{Dim}(\Theta_{X_i|P_i}) \quad (2)$$

with $N_{ijk}$ the occurrence number of $\{X_i = x_k$ and $P_i = pa_j\}$ in $\mathcal{D}$.

The principle of the MWST algorithm is rather different. This algorithm determines the best tree that links all the variables, using a mutual information measurement like in (Chow and Liu, 1968) or the BIC score variation when two variables become linked as proposed by (Heckerman et al., 1995). The aim is to find an optimal solution, but in a space limited to trees.

[1] As (Friedman, 1997), we consider that the BIC/MDL score is a function of the graph $\mathcal{G}$ and the parameters $\Theta$, generalizing the classical definition of the BIC score which is defined with our notation by $BIC(\mathcal{G},\Theta^*)$ where $\Theta^*$ is obtained by maximizing the likelihood or $BIC(\mathcal{G},\Theta)$ score for a given $\mathcal{G}$.

### 2.2 Bayesian classifiers

Bayesian classifiers as Naive Bayes have shown excellent performances on many datasets. Even if the Naive Bayes classifier has underlying heavy independence assumptions, (Domingos and Pazzani, 1997) have shown that it is optimal for conjunctive and disjunctive concepts. They have also shown that the Naive Bayes classifier does not require attribute independence to be optimal under Zero-One loss.

Augmented Naive Bayes classifier appear as a natural extension to the Naive Bayes classifier. It allows to relax the assumption of independence of attributes given the class variable. Many ways to find the best tree to augment the Naive Bayes classifier have been studied. These Tree Augmented Naive Bayes classifiers (Geiger, 1992; Friedman et al., 1997) are a restricted family of Bayesian Networks in which the class variable has no parent and each other attribute has as parents the class variable and at most one other attribute. The BIC score of such a Bayesian network is given by Eqn.3.

$$BIC(\mathcal{T}_{AN},\Theta) = bic(C,\emptyset,\Theta_{C|\emptyset}) + \sum_i bic(X_i,\{C,P_i\},\Theta_{X_i|\{C,P_i\}}) \quad (3)$$

where $C$ stands for the class node and $P_i$ could only be the emptyset $\emptyset$ or a singleton $\{X_j\}$, $X_j \notin \{C, X_i\}$.

Forest Augmented Naive Bayes classifier (FAN) is very close to the TAN one. In this model, the augmented structure is not a tree, but a set of disconnected trees in the attribute space (Sacha, 1999).

### 2.3 Dealing with incomplete data

#### 2.3.1 Practical issue

Nowadays, more and more datasets are available, and most of them are incomplete. When we want to build a model from an incomplete dataset, it is often possible to consider only the complete samples in the dataset. But, in this case, we do not have a lot of data to learn the model. For instance, if we have a dataset with 2000 samples on 20 attributes with a probability of 20% that a data is missing, then, only

23 samples (in average) are complete. Generalizing from the example, we see that we cannot ignore the problem of incomplete datasets.

### 2.3.2 Nature of missing data

Let $\mathcal{D} = \{X_i^l\}_{1 \leqslant i \leqslant n, 1 \leqslant l \leqslant N}$ our dataset, with $\mathcal{D}_o$ the observed part of $\mathcal{D}$, $\mathcal{D}_m$ the missing part and $\mathcal{D}_{co}$ the set of completely observed cases in $\mathcal{D}_o$. Let also $\mathcal{M} = \{M_{il}\}$ with $M_{il} = 1$ if $X_i^l$ is missing, 0 if not. We then have the following relations:

$$\begin{aligned}
\mathcal{D}_m &= \{X_i^l \ / \ M_{il} = 1\}_{1 \leqslant i \leqslant n, 1 \leqslant l \leqslant N} \\
\mathcal{D}_o &= \{X_i^l \ / \ M_{il} = 0\}_{1 \leqslant i \leqslant n, 1 \leqslant l \leqslant N} \\
\mathcal{D}_{co} &= \{[X_1^l \dots X_n^l] \ / \ [M_{1l} \dots M_{nl}] = [0 \dots 0]\}_{1 \leqslant l \leqslant N}
\end{aligned}$$

Dealing with missing data depends on their nature. (Rubin, 1976) identified several types of missing data:

- MCAR (Missing Completly At Random): $P(\mathcal{M}|\mathcal{D}) = P(\mathcal{M})$, the probability for data to be missing does not depend on $\mathcal{D}$,
- MAR (Missing At Random): $P(\mathcal{M}|\mathcal{D}) = P(\mathcal{M}|\mathcal{D}_o)$, the probability for data to be missing depends on observed data,
- NMAR (Not Missing At Random): the probability for data to be missing depends on both observed and missing data.

MCAR and MAR situations are the easiest to solve as observed data include all necessary information to estimate missing data distribution. The case of NMAR is trickier as outside information has to be used to model the missing data distribution.

### 2.3.3 Learning $\Theta$ with incomplete data

With MCAR data, the first and simplest possible approach is the *complete case analysis*. This is a parameter estimation based on $\mathcal{D}_{co}$, the set of completely observed cases in $\mathcal{D}_o$. When $\mathcal{D}$ is MCAR, the estimator based on $\mathcal{D}_{co}$ is unbiased. However, with a high number of variables the probability for a case $[X_1^l \dots X_n^l]$ to be completely measured is low and $\mathcal{D}_{co}$ may be empty.

One advantage of Bayesian networks is that, if only $X_i$ and $P_i = Pa(X_i)$ are measured, then the corresponding conditional probability table can be estimated. Another possible method with MCAR cases is the *available case analysis*, *i.e.* using for the estimation of each conditional probability $P(X_i|Pa(X_i))$ the cases in $\mathcal{D}_o$ where $X_i$ and $Pa(X_i)$ are measured, not only in $\mathcal{D}_{co}$ (where all $X_i$'s are measured) as in the previous approach.

Many methods try to rely more on all the observed data. Among them are *sequential updating* (Spiegelhalter and Lauritzen, 1990), *Gibbs sampling* (Geman and Geman, 1984), and *expectation maximisation* (EM) in (Dempster et al., 1977). Those algorithms use the missing data MAR properties. More recently, *bound and collapse* algorithm (Ramoni and Sebastiani, 1998) and *robust Bayesian estimator* (Ramoni and Sebastiani, 2000) try to resolve this task whatever the nature of missing data.

EM has been adapted by (Lauritzen, 1995) to Bayesian network parameter learning when the structure is known. Let $\log P(\mathcal{D}|\Theta) = \log P(\mathcal{D}_o, \mathcal{D}_m|\Theta)$ be the data log-likelihood. $\mathcal{D}_m$ being an unmeasured random variable, this log-likelihood is also a random variable function of $\mathcal{D}_m$. By establishing a reference model $\Theta^*$, it is possible to estimate the probability density of the missing data $P(\mathcal{D}_m|\Theta^*)$ and therefore to calculate $Q(\Theta : \Theta^*)$, the expectation of the previous log-likelihood:

$$Q(\Theta : \Theta^*) = E_{\Theta^*}\left[\log P(\mathcal{D}_o, \mathcal{D}_m|\Theta)\right] \tag{4}$$

So $Q(\Theta : \Theta^*)$ is the expectation of the likelihood of any set of parameters $\Theta$ calculated using a distribution of the missing data $P(\mathcal{D}_m|\Theta^*)$. This equation can be re-written as follows.

$$Q(\Theta : \Theta^*) = \sum_{i=1}^{n} \sum_{X_i = x_k} \sum_{P_i = pa_j} N_{ijk}^* \log \theta_{ijk} \tag{5}$$

where $N_{ijk}^* = E_{\Theta^*}[N_{ijk}] = N \times P(X_i = x_k, P_i = pa_j|\Theta^*)$is obtained by inference in the network $< \mathcal{G}, \Theta^* >$ if the $\{X_i, P_i\}$ are not completely measured, or else only by mere counting.

(Dempster et al., 1977) proved convergence of the EM algorithm, as the fact that it was not necessary to find the global optimum $\Theta^{i+1}$ of function $Q(\Theta : \Theta^i)$ but simply a value which would increase function $Q$ (*Generalized* EM).

### 2.3.4 Learning $\mathcal{G}$ with incomplete dataset

The main methods for structural learning with incomplete data use the EM principle: *Alternative Model Selection* EM (AMS-EM) pro-

posed by (Friedman, 1997) or *Bayesian Structural* EM (BS-EM) (Friedman, 1998). We can also cite the *Hybrid Independence Test* proposed in (Dash and Druzdzel, 2003) that can use EM to estimate the essential sufficient statistics that are then used for an independence test in a constraint-based method. (Myers et al., 1999) also proposes a structural learning method based on genetic algorithm and MCMC. We will now explain the structural EM algorithm principle in details and see how we could adapt it to learn a TAN model.

## 3 Structural em algorithm

### 3.1 General principle

The EM principle, which we have described above for parameter learning, applies more generally to structural learning (Algorithm 1 as proposed by (Friedman, 1997; Friedman, 1998)).

**Algorithm 1** : Generic EM for structural learning

1: **Init:** $i = 0$
Random or heuristic choice of the initial Bayesian network $(\mathcal{G}^0, \Theta^0)$
2: **repeat**
3: $i = i + 1$
4: $(\mathcal{G}^i, \Theta^i) = \underset{\mathcal{G},\Theta}{\text{argmax}} \quad Q(\mathcal{G}, \Theta : \mathcal{G}^{i-1}, \Theta^{i-1})$
5: **until** $|Q(\mathcal{G}^i, \Theta^i : \mathcal{G}^{i-1}, \Theta^{i-1}) - Q(\mathcal{G}^{i-1}, \Theta^{i-1} : \mathcal{G}^{i-1}, \Theta^{i-1})| \leqslant \epsilon$

The maximization step in this algorithm (step 4) has to be performed in the joint space $\{\mathcal{G}, \Theta\}$ which amounts to searching the best structure and the best parameters corresponding to this structure. In practice, these two steps are clearly distinct[2]:

$$\mathcal{G}^i = \underset{\mathcal{G}}{\text{argmax}} \quad Q(\mathcal{G}, \bullet : \mathcal{G}^{i-1}, \Theta^{i-1}) \tag{6}$$

$$\Theta^i = \underset{\Theta}{\text{argmax}} \quad Q(\mathcal{G}^i, \Theta : \mathcal{G}^{i-1}, \Theta^{i-1}) \tag{7}$$

where $Q(\mathcal{G}, \Theta : \mathcal{G}^*, \Theta^*)$ is the expectation of the likelihood of any Bayesian network $< \mathcal{G}, \Theta >$ computed using a distribution of the missing data $P(\mathcal{D}_m | \mathcal{G}^*, \Theta^*)$.

Note that the first search (Eqn.6) in the space of possible graphs takes us back to the initial problem, i.e. the search for the best structure in a super-exponential space. However, with *Generalised* EM it is sufficient to look for a better solution rather than the best possible one, without affecting the algorithm convergence properties. This search for a better solution can then be done in a limited space, like for example $\mathcal{V}_\mathcal{G}$, the set of the neigbours of graph $\mathcal{G}$ that have been generated by removal, addition or inversion of an arc.

[2]The notation $Q(\mathcal{G}, \bullet : \dots)$ used in Eqn.6 stands for $E_\Theta[Q(\mathcal{G}, \Theta : \dots)]$ for Bayesian scores or $Q(\mathcal{G}, \Theta^o : \dots)$ where $\Theta^o$ is obtained by likelihood maximisation.

**Algorithm 2** : Detailed EM for structural learning

1: **Init:** $finished = false$, $i = 0$
Random or heuristic choice of the initial Bayesian network $(\mathcal{G}^0, \Theta^{0,0})$
2: **repeat**
3: $j = 0$
4: **repeat**
5: $\Theta^{i,j+1} = \underset{\Theta}{\text{argmax}} \quad Q(\mathcal{G}^i, \Theta : \mathcal{G}^i, \Theta^{i,j})$
6: $j = j + 1$
7: **until** convergence $(\Theta^{i,j} \rightarrow \Theta^{i,j^o})$
8: **if** $i = 0$ or $|Q(\mathcal{G}^i, \Theta^{i,j^o} : \mathcal{G}^{i-1}, \Theta^{i-1,j^o}) - Q(\mathcal{G}^{i-1}, \Theta^{i-1,j^o} : \mathcal{G}^{i-1}, \Theta^{i-1,j^o})| > \epsilon$ **then**
9: $\mathcal{G}^{i+1} = \arg\max_{\mathcal{G} \in \mathcal{V}_{\mathcal{G}^i}} \quad Q(\mathcal{G}, \bullet : \mathcal{G}^i, \Theta^{i,j^o})$
10: $\Theta^{i+1,0} = \underset{\Theta}{\text{argmax}} \quad Q(\mathcal{G}^{i+1}, \Theta : \mathcal{G}^i, \Theta^{i,j^o})$
11: $i = i + 1$
12: **else**
13: $finished = true$
14: **end if**
15: **until** $finished$

Concerning the search in the space of the parameters (Eqn.7), (Friedman, 1997) proposes repeating the operation several times, using a clever initialisation. This step then amounts to running the parametric EM algorithm for each structure $\mathcal{G}^i$, starting with structure $\mathcal{G}^0$ (steps 4 to 7 of Algorithm 2). The two structural EM algorithms proposed by Friedman can therefore be considered as greedy search algorithms, with EM parameter learning at each iteration.

### 3.2 Choice of function $Q$

We now have to choose the function $Q$ that will be used for structural learning. The likelihood used for parameter learning is not a good indicator to determine the best graph since it gives more importance to strongly connected structures. Moreover, it is impossible to compute marginal likelihood when data are incomplete, so that it is necessary to rely on an efficient approximation like those reviewed by (Chicker-

ing and Heckerman, 1996). In complete data cases, the most frequently used measurements are the BIC/MDL score and the Bayesian BDe score (see paragraph 2.1). When proposing the MS-EM and MWST-EM algorithms, (Friedman, 1997) shows how to use the BIC/MDL score with incomplete data, by applying the principle of Eqn.4 to the BIC score (Eqn.1) instead of likelihood. Function $Q^{BIC}$ is defined as the BIC score expectation by using a certain probability density on the missing data $P(\mathcal{D}_m|\mathcal{G}^*,\Theta^*)$ :

$$Q^{BIC}(\mathcal{G},\Theta:\mathcal{G}^*,\Theta^*) = \quad (8)$$
$$E_{\mathcal{G}^*,\Theta^*}\left[\log P(\mathcal{D}_o,\mathcal{D}_m|\mathcal{G},\Theta)\right] - \frac{1}{2}\mathrm{Dim}(\mathcal{G})\log N$$

As the BIC score is decomposable, so is $Q^{BIC}$.

$$Q^{BIC}(\mathcal{G},\Theta:\mathcal{G}^*,\Theta^*) = \sum_i Q^{bic}(X_i,P_i,\Theta_{Xi|P_i}:\mathcal{G}^*,\Theta^*)$$

where $\quad Q^{bic}(X_i,P_i,\Theta_{Xi|P_i}:\mathcal{G}^*,\Theta^*) = \quad (9)$

$$\sum_{X_i=x_k}\sum_{P_i=pa_j} N^*_{ijk}\log\theta_{ijk} - \frac{\log N}{2}\mathrm{Dim}(\Theta_{Xi|P_i}) \quad (10)$$

with $N^*_{ijk} = E_{\mathcal{G}^*,\Theta^*}[N_{ijk}] = N * P(X_i = x_k, P_i = pa_j|\mathcal{G}^*,\Theta^*)$ obtained by inference in the network $\{\mathcal{G}^*,\Theta^*\}$ if $\{X_i,P_i\}$ are not completely measured, or else only by mere counting. With the same reasoning, (Friedman, 1998) proposes the adaptation of the BDe score to incomplete data.

## 4 TAN-EM, a structural EM for classification

(Leray and François, 2005) have introduced MWST-EM an adaptation of MWST dealing with incomplete datasets. The approach we propose here is using the same principles in order to efficiently learn TAN classifiers from incomplete datasets.

### 4.1 MWST-EM, a structural EM in the space of trees

Step 1 of Algorithm 2, like all the previous algorithms, deals with the choice of the initial structure. The choice of an oriented chain graph linking all the variables proposed by (Friedman, 1997) seems even more judicious here, since this chain graph also belongs to the tree space. Steps 4 to 7 do not change. They deal with the running of the parametric EM algorithm for each structure $\mathcal{B}^i$, starting with structure $\mathcal{B}^0$.

There is a change from the regular structural EM algorithm in step 9, i.e. the search for a better structure for the next iteration. With the previous structural EM algorithms, we were looking for the best DAG among the neighbours of the current graph. With MWST-EM, we can directly get the best tree that maximises function $Q$.

In paragraph 2.1, we briefly recalled that the MWST algorithm used a similarity function between two nodes which was based on the BIC score variation whether $X_j$ is linked to $X_i$ or not. This function can be summed up in the following (symmetrical) matrix:

$$\left[M_{ij}\right]_{1\leqslant i,j\leqslant n} = \left[bic(X_i,X_j,\Theta_{X_i|X_j}) - bic(X_i,\emptyset,\Theta_{X_i})\right] \quad (11)$$

where the local *bic* score is defined in Eqn.2.

Running maximum (weight) spanning algorithms like Kruskal's on matrix $M$ enables us to obtain the best tree $\mathcal{T}$ that maximises the sum of the local scores on all the nodes, i.e. function $BIC$ of Eqn.2.

By applying the principle we described in section 3.2, we can then adapt MWST to incomplete data by replacing the local *bic* score of Equn.11 with its expectation; to do so, we use a certain probability density of the missing data $P(\mathcal{D}_m|\mathcal{T}^*,\Theta^*)$ :

$$\left[M^Q_{ij}\right]_{i,j} = \left[Q^{bic}(X_i,P_i=\{X_j\},\Theta_{X_i|X_j}:\mathcal{T}^*,\Theta^*) - Q^{bic}(X_i,P_i=\emptyset,\Theta_{X_i}:\mathcal{T}^*,\Theta^*)\right] \quad (12)$$

With the same reasoning, running a maximum (weight) spanning tree algorithm on matrix $M^Q$ enables us to get the best tree $\mathcal{T}$ that maximises the sum of the local scores on all the nodes, i.e. function $Q^{BIC}$ of Eqn.9.

### 4.2 TAN-EM, a structural EM for classification

The score used to find the best TAN structure is very similar to the one used in MWST, so we can adapt it to incomplete datasets by defining the following score matrix:

$$\left[M^Q_{ij}\right]_{i,j} = \left[Q^{bic}(X_i,P_i=\{C,X_j\},\Theta_{X_i|X_jC}:\mathcal{T}^*,\Theta^*) - Q^{bic}(X_i,P_i=\{C\},\Theta_{X_i|C}:\mathcal{T}^*,\Theta^*)\right] \quad (13)$$

Using this new score matrix, we can use the approach previously proposed for MWST-EM to

get the best augmented tree, and connect the class node to all the other nodes to obtain the TAN structure. We are currently using the same reasoning to find the best forest "extension".

### 4.3 Related works

(Meila-Predoviciu, 1999) applies MWST algorithm and EM principle, but in another framework, learning mixtures of trees. In this work, the data is complete, but a new variable is introduced in order to take into account the weight of each tree in the mixture. This variable isn't measured so EM is used to determine the corresponding parameters.

(Peña et al., 2002) propose a change inside the framework of the SEM algorithm resulting in an alternative approach for learning Bayes Nets for clustering more efficiently.

(Greiner and Zhou, 2002) propose maximizing conditional likelihood for BN parameter learning. They apply their method to MCAR incomplete data by using *available case analysis* in order to find the best TAN classifier.

(Cohen et al., 2004) deal with TAN classifiers and EM principle for partially unlabeled data. In there work, only the variable corresponding to the class can be partially missing whereas any variable can be partially missing in our TAN-EM extension.

## 5 Experiments

### 5.1 Protocol

The experiment stage aims at evaluating the Tree Augmented Naive Bayes classifier on incomplete datasets from UCI repository[3]: `Hepatitis`, `Horse`, `House`, `Mushrooms` and `Thyroid`.

The TAN-EM method we proposed here is compared to the Naive Bayes classifier with EM parameters learning. We also indicate the classification rate obtained by three methods: MWST-EM, SEM initialised with a random chain and SEM initialised with the tree given by MWST-EM (SEM+T). The first two methods are dedicated to classification tasks while the others do not consider the class node as a specific variable.

We also give an $\alpha$ confidence interval for each classification rate, based on Eqn.14 proposed by (Bennani and Bossaert, 1996):

$$I(\alpha, N) = \frac{T + \frac{Z_\alpha^2}{2N} \pm Z_\alpha \sqrt{\frac{T(1-T)}{N} + \frac{Z_\alpha^2}{4N^2}}}{1 + \frac{Z_\alpha^2}{N}} \qquad (14)$$

where $N$ is the number of samples in the dataset, $T$ is the classification rate and $Z_\alpha = 1,96$ for $\alpha = 95\%$.

### 5.2 Results

The results are summed up in Table 1. First, we could see that even if the Naive Bayes classifier often gives good results, the other tested methods allow to obtain better classification rates. But, where all runnings of NB-EM give the same results, as EM parameter learning only needs an initialisiation, the other methods do not always give the same results, and then, the same classification rates. We have also noticed (not reported here) that, excepting NB-EM, TAN-EM seems the most stable method concerning the evaluated classification rate while MWST-EM seems to be the less stable.

The method MWST-EM can obtain very good structures with a good initialisation. Then, initialising it with the results of MWST-EM gives us stabler results (see (Leray and François, 2005) for a more specific study of this point).

In our tests, except for this `house` dataset, TAN-EM always obtains a structure that lead to better classification rates in comparison with the other structure learning methods.

Surprisingly, we also remark that MWST-EM can give good classification rates even if the class node is connected to a maximum of two other attributes.

Regarding the log-likelihood reported in Table 1, we see that the TAN-EM algorithm finds structures that can also lead to a good approximation of the underlying probability distribution of the data, even with a strong constraint on the graph structure.

Finally, the Table 1 illustrates that TAN-EM and MWST-EM have about the same complexity (regarding the computational time) and

[3] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

| Datasets | N | learn | test | #C | %I | NB-EM | MWST-EM | TAN-EM | SEM | SEM+T |
|---|---|---|---|---|---|---|---|---|---|---|
| Hepatitis | 20 | 90 | 65 | 2 | 8.4 | 70.8 [58.8;80.5] | 73.8 [62.0;83.0] | **75.4** [63.6;84.2] | 66.1 [54.0;76.5] | 66.1 [54.0;76.5] |
| | | | | | | -1224.2 ; 29.5 | **-1147.6** ; 90.4 | **-1148.7** ; 88.5 | -1211.5 ; 1213.1 | -1207.9 ; 1478.5 |
| Horse | 28 | 300 | 300 | 2 | 88.0 | 75 [63.5;83.8] | 77.9 [66.7;86.2] | **80.9** [69.9;88.5] | 66.2 [54.3;76.3] | 66.2 [54.3;76.3] |
| | | | | | | -5589.1 ; 227.7 | **-5199.6** ; 656.1 | -5354.4 ; 582.2 | -5348.3 ; 31807 | -5318.2 ; 10054 |
| House | 17 | 290 | 145 | 2 | 46.7 | 89.7 [83.6;93.7] | **93.8** [88.6;96.7] | 92.4 [86.9;95.8] | 92.4 [86.9;95.8] | **93.8** [88.6;96.7] |
| | | | | | | -2203.4 ; 110.3 | -2518.0 ; 157.0 | **-2022.2** ; 180.7 | -2524.4 ; 1732.4 | -2195.8 ; 3327.2 |
| Mushrooms | 23 | 5416 | 2708 | 2 | 30.5 | **92.8** [91.7;93.8] | 74.7 [73.0;73.4] | 91.3 [90.2;92.4] | 74.9 [73.2;76.5] | 74.9 [73.2;76.5] |
| | | | | | | -97854 ; 2028.9 | -108011 ; 6228.2 | **-87556** ; 5987.4 | -111484 ; 70494 | -110828 ; 59795 |
| Thyroid | 22 | 2800 | 972 | 2 | 29.9 | 95.3 [93.7;96.5] | 93.8 [92.1;95.2] | **96.2** [94.7;97.3] | 93.8 [92.1;95.2] | 93.8 [92.1;95.2] |
| | | | | | | -39348 ; 1305.6 | -38881 ; 3173.0 | **-38350** ; 3471.4 | **-38303** ; 17197 | -39749 ; 14482 |

Table 1: First line: best classification rate (on 10 runs, except Mushrooms on 5, in %) on test dataset and its confidence interval, for the following learning algorithms: NB-EM, MWST-EM, SEM, TAN-EM and SEM+T. Second line: log-likelihood estimated with test data and calculation time (sec) for the network with the best classification rate. The first six columns give us the name of the dataset and some of its properties : number of attributes, learning sample size, test sample size, number of classes and percentage of incomplete samples.

are a good compromise between NB-EM (classical Naive Bayes with EM parameter learning) and MWST-EM (greedy search with incomplete data).

## 6 Conclusions and prospects

Bayesian networks are a tool of choice for reasoning in uncertainty, with incomplete data. However, most of the time, Bayesian network structural learning only deal with complete data. We have proposed here an adaptation of the learning process of Tree Augmented Naive Bayes classifier from incomplete datasets (and not only partially labelled data). This method has been successfuly tested on some datasets.

We have seen that TAN-EM was a good classification tool compared to other Bayesian networks we could obtained with structural EM like learning methods.

Our method can easily be extended to unsupervised classification tasks by adding a new step in order to determine the best cardinality for the class variable.

Related future works are the adaptation of some other Augmented Naive Bayes classifiers for incomplete datasets (FAN for instance), but also the study of these methods with MAR datasets.

MWST-EM, TAN-EM and SEM methods are respective adaptations of MWST, TAN and greedy search to incomplete data. These algorithms are applying in (subspace of) DAG space. (Chickering and Meek, 2002) proposed an optimal search algorithm (GES) which deals with Markov equivalent space. Logically enough, the next step in our research is to adapt GES to incomplete datasets. Then we could test results of this method on classification tasks.

## 7 Acknowledgement

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

## References

Y. Bennani and F. Bossaert. 1996. Predictive neural networks for traffic disturbance detection in the telephone network. In *Proceedings of IMACS-CESA'96*, page xx, Lille, France.

D. Chickering and D. Heckerman. 1996. Efficient Approximation for the Marginal Likelihood of Incomplete Data given a Bayesian Network. In *UAI'96*, pages 158–168. Morgan Kaufmann.

D. Chickering and C. Meek. 2002. Finding optimal bayesian networks. In Adnan Darwiche and Nir Friedman, editors, *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 94–102, S.F., Cal. Morgan Kaufmann Publishers.

D. Chickering, D. Geiger, and D. Heckerman. 1995. Learning bayesian networks: Search methods and experimental results. In *Proceedings of Fifth Conference on Artificial Intelligence and Statistics*, pages 112–128.

C.K. Chow and C.N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

I. Cohen, F. G. Cozman, N. Sebe, M. C. Cirelo, and T. S. Huang. 2004. Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(12):1553–1568.

D. Dash and M.J. Druzdzel. 2003. Robust independence testing for constraint-based learning of causal structure. In *Proceedings of The Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*, pages 167–174.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39:1–38.

P. Domingos and M. Pazzani. 1997. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130.

N. Friedman, D. Geiger, and M. Goldszmidt. 1997. bayesian network classifiers. *Machine Learning*, 29(2-3):131–163.

N. Friedman. 1997. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the 14th International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann.

N. Friedman. 1998. The bayesian structural EM algorithm. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 129–138, San Francisco, July. Morgan Kaufmann.

D. Geiger. 1992. An entropy-based learning algorithm of bayesian conditional trees. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference (UAI-1992)*, pages 92–97, San Mateo, CA. Morgan Kaufmann Publishers.

S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November.

R. Greiner and W. Zhou. 2002. Structural extension to logistic regression. In *Proceedings of the Eighteenth Annual National Conference on Artificial Intelligence (AAI02)*, pages 167–173, Edmonton, Canada.

D. Heckerman, D. Geiger, and M. Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

S. Lauritzen. 1995. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201.

P. Leray and O. François. 2005. bayesian network structural learning and incomplete data. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR 2005), Espoo, Finland*, pages 33–40.

M. Meila-Predoviciu. 1999. *Learning with Mixtures of Trees*. Ph.D. thesis, MIT.

J.W. Myers, K.B. Laskey, and T.S. Lewitt. 1999. Learning bayesian network from incomplete data with stochatic search algorithms. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI99)*.

J.M. Peña, J. Lozano, and P. Larrañaga. 2002. Learning recursive bayesian multinets for data clustering by means of constructive induction. *Machine Learning*, 47:1:63–90.

M. Ramoni and P. Sebastiani. 1998. Parameter estimation in Bayesian networks from incomplete databases. *Intelligent Data Analysis*, 2:139–160.

M. Ramoni and P. Sebastiani. 2000. Robust learning with missing data. *Machine Learning*, 45:147–170.

D.B. Rubin. 1976. Inference and missing data. *Biometrika*, 63:581–592.

J.P. Sacha. 1999. *New Synthesis of bayesian Network Classifiers and Cardiac SPECT Image Interpretation*. Ph.D. thesis, The University of Toledo.

D. J. Spiegelhalter and S. L. Lauritzen. 1990. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605.

# Lattices for Studying Monotonicity of Bayesian Networks

Linda C. van der Gaag, Silja Renooij, and Petra L. Geenen
Department of Information and Computing Sciences, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands.
e-mail: {linda,silja,petrag}@cs.uu.nl

**Abstract**

In many real problem domains, the main variable of interest behaves monotonically in terms of the observable variables, in the sense that higher values for the variable of interest become more likely with higher-ordered observations. Unfortunately, establishing whether or not a Bayesian network exhibits these monotonicity properties is highly intractable in general. In this paper, we present a method that, by building upon the concept of assignment lattice, provides for identifying any violations of the properties of (partial) monotonicity of the output and for constructing minimal offending contexts. We illustrate the application of our method with a real Bayesian network in veterinary science.

## 1 Introduction

In many problem domains, the variables of importance have different roles. Often, a number of observable input variables and a single output variable are distinguished. In a biomedical diagnostic application, for example, the input variables capture the findings from different diagnostic tests and the output variable models the possible diseases. Multiple input variables and a single output variable in fact are typically found in any type of diagnostic problem.

For many problems, the relation between the output variable and the observable input variables is monotone in the sense that higher-ordered values for the input variables give rise to a higher-ordered output for the main variable of interest. In a biomedical diagnostic application, for example, observing symptoms and signs that are more severe will result in a more severe disease being the most likely value of the diagnostic variable. The concept of monotonicity in distribution has been introduced to capture this type of knowledge for Bayesian networks (Van der Gaag *et al.*, 2004). More specifically, a network is said to be isotone if the conditional probability distribution computed for the output variable given specific observations is stochastically dominated by any such distribution given higher-ordered observations.

Unfortunately, the problem of verifying whether or not a Bayesian network exhibits the properties of monotonicity from its domain of application is highly intractable in general and remains to be so even for polytrees. Although an approximate anytime algorithm is available for deciding if a given network is monotone, we found that its runtime requirements tend to forestall use in a practical setting.

In this paper, we present a new, more practicable method for studying monotonicity of Bayesian networks. The method builds upon a lattice of all joint value assignments to the observable variables under study; this lattice moreover, is enhanced with information about the effects of the various assignments on the probability distribution over the main variable of interest. The assignment lattice is used for identifying any violations of the properties of monotonicity in the network. Subsequently, minimal offending contexts are constructed that characterise the identified violations and provide for further investigation.

The runtime complexity of our method is exponential in the number of observable variables under study. For larger networks including a large number of observable variables therefore, our method provides for verifying partial mono-

tonicity for limited subsets of variables only. The unfavourable computational complexity of the problem, however, is likely to forestall the design of essentially more efficient methods.

We applied our verification method for studying monotonicity to a Bayesian network in veterinary science. In recent years, we developed a network for the detection of classical swine fever in pigs. Both the network's structure and its associated probabilities were elicited from two experts. During the elicitation interviews, the experts had produced several statements that suggested properties of monotonicity. We studied these properties in the network using our verification method. We found a small number of violations of the suggested properties of monotonicity, which proved to be indicative of modelling inadequacies.

The paper is organised as follows. In Section 2, we review the concept of monotonicity. In Section 3, we present our method for studying monotonicity. We report on the application of our method in Section 4. The paper ends with our concluding observations in Section 5.

## 2 The Concept of Monotonicity

Upon reviewing the concept of monotonicity, we assume that a Bayesian network under study includes a single output variable $C$ and $n$ observable variables $E_i$, $i = 1, \ldots, n$, $n \geq 1$; in addition, the network may include an arbitrary number of intermediate variables which are not observed in practice. Each variable $X_i$ in the network adopts one of a finite set $\Omega(X_i) = \{x_i^1, \ldots, x_i^m\}$, $m \geq 1$, of values. We assume that there exists a total ordering $\leq$ on this set of values; without loss of generality, we assume that $x_i^j \leq x_i^k$ whenever $j \leq k$. The orderings per variable are taken to induce a partial ordering $\preceq$ on the set of joint value assignments to any subset of the network's variables.

The concept of *monotonicity in distribution* now builds upon the posterior probability distributions over the output variable given the various joint value assignments to the network's observable variables (Van der Gaag *et al.*, 2004). The concept, more specifically, is defined in terms of stochastic dominance among these distributions. For a probability distribution $\Pr(C)$ over the output variable, the cumulative distribution function $F_{\Pr}$ is defined as $F_{\Pr}(c^i) = \Pr(C \leq c^i)$ for all values $c^i$ of $C$. For two distributions $\Pr(C)$ and $\Pr'(C)$ over $C$, associated with $F_{\Pr}(C)$ and $F_{\Pr'}(C)$ respectively, we say that $\Pr'(C)$ is *stochastically dominant* over $\Pr(C)$, denoted $\Pr(C) \leq \Pr'(C)$, if $F_{\Pr'}(c^i) \leq F_{\Pr}(c^i)$ for all $c^i \in \Omega(C)$ (Berger, 1980). We now say that the network is *isotone* in its set of observable variables $E$ if

$$e \preceq e' \rightarrow \Pr(C \mid e) \leq \Pr(C \mid e')$$

for all joint value assignments $e, e'$ to $E$. Informally speaking, we have that a network is isotone in distribution if entering a higher-ordered value assignment to the observable variables cannot make higher-ordered values of the output variable less likely. The concept of antitonicity has the reverse interpretation: the network is said to be *antitone* in $E$ if

$$e \preceq e' \rightarrow \Pr(C \mid e) \geq \Pr(C \mid e')$$

for all value assignments $e, e'$ to $E$. Note that if a network is isotone in its observable variables given the orderings $\leq$ on their sets of values, then the network is antitone given the reversed orderings. Although antitonicity thus is (reversely) equivalent to isotonicity, we explicitly distinguish between the two types of monotonicity since a domain of application may exhibit an intricate combination of isotonicity and antitonicity for interrelated observable variables.

Building upon the above definitions, we have that deciding whether or not a Bayesian network is isotone amounts to verifying that entering any higher-ordered value assignment to its observable variables results in a stochastically dominant probability distribution over the output variable. Establishing antitonicity amounts to verifying that entering any such assignment results in a dominated distribution.

The problem of deciding monotonicity for a Bayesian network is known to be coNP$^{\text{PP}}$-complete in general, and in fact remains coNP-complete for polytrees (Van der Gaag *et al.*,

2004). In view of these complexity considerations, Van der Gaag *et al.* designed an approximate anytime algorithm for verifying whether or not a given network is monotone. This algorithm studies the relation between the output variable and each observable variable separately, in terms of the sign of the qualitative influence between them (Wellman, 1990); upon inconclusive results, iteratively tightened numerical bounds are established on the relevant probabilities using an anytime method available from Liu and Wellman (1998). Unfortunately, the overall runtime requirements of the algorithm tend to forestall use in a practical setting.

## 3 Studying Monotonicity

Our method for studying monotonicity in Bayesian networks now builds upon the construct of an assignment lattice. The lattice includes all joint value assignments to the set of observable variables and is enhanced with probabilistic information computed from the network under study. From the lattice, all violations of the properties of monotonicity are identified, from which minimal offending contexts are constructed for further investigation.

### 3.1 The assignment lattice

The *assignment lattice* for the set $E$ of observable variables of a Bayesian network captures all joint value assignments to $E$, along with the partial ordering between them. For each value assignment $e$ to $E$, an element $L(e)$ is included in the lattice. The bottom of the lattice encodes the assignment $e$ for which we have that $e \preceq e'$ for all $e' \in \Omega(E)$; the bottom thus encodes the lowest-ordered value assignment to $E$. The top of the lattice encodes the assignment $e''$ for which we have that $e' \preceq e''$ for all $e' \in \Omega(E)$; the top thus encodes the highest-ordered value assignment to $E$. In the lattice, we further have that an element $L(e)$ precedes an element $L(e'')$ if $e \preceq e''$; we moreover say that $L(e)$ precedes $L(e'')$ directly if there is no assignment $e'$ with $e \prec e' \prec e''$. The partial ordering defined by the lattice thus coincides with the partial ordering $\preceq$ on the joint value assignments to $E$. Figure 1 depicts, as an example, the assignment lattice

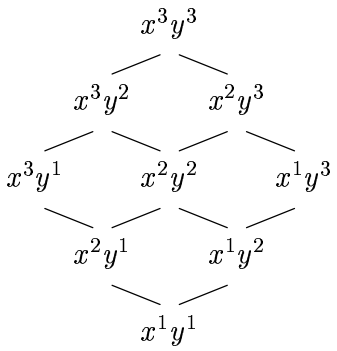



Figure 1: An example assignment lattice for two ternary observable variables.

for the two ternary variables $X$ and $Y$. For further information about lattices in general, we refer to (Grätzer, 1971).

To describe the effects of the various value assignments $e$ on the probability distribution over the output variable, the assignment lattice is enhanced with probabilistic information. With each element $L(e)$ of the lattice, the conditional probability distribution $\Pr(C \mid e)$ over the output variable $C$ is associated. We note that these distributions are readily computed from the Bayesian network under study. We will return to the complexity of the computations involved in Section 3.4.

### 3.2 Exploiting the assignment lattice

By building upon the assignment lattice, verifying monotonicity in distribution amounts to checking whether particular dominance properties hold among the probability distributions associated with the lattice's elements.

We recall that a Bayesian network is isotone in its set of observable variables $E$ if entering a higher-ordered value assignment to $E$ results in a stochastically dominant probability distribution over the output variable $C$. We further recall that the partial ordering $\preceq$ on the joint value assignments to $E$ is encoded directly in the assignment lattice for $E$. We now consider all pairs of conditional probability distributions $\Pr(C \mid e)$ and $\Pr(C \mid e')$ associated with elements $L(e)$ and $L(e')$ in the lattice such that $L(e)$ precedes $L(e')$. If for each such pair we have that $\Pr(C \mid e) \leq \Pr(C \mid e')$, then the net-

work is isotone in $E$. The network is antitone in $E$ if for each such pair of distributions we have that $\Pr(C \mid e') \leq \Pr(C \mid e)$. We note that, since the property of stochastic dominance is transitive, we have to study the dominance properties of the distributions of directly linked pairs of elements in the lattice only to decide upon isotonicity or antitonicity.

As mentioned in Section 2, a domain of application may exhibit an intricate combination of properties of isotonicity and antitonicity for interrelated observable variables. As an example, we consider the two variables $X$ and $Y$ such that the output variable of interest behaves isotonically in $X$ and antitonically in $Y$. We focus on the value assignments

$$\begin{aligned} e_{ij} &\equiv x^i y^{j+1} \\ e'_{ij} &\equiv x^{i+1} y^j \\ e''_{ij} &\equiv x^{i+1} y^{j+1} \end{aligned}$$

to these variables. Note that for the three assignments we have that $e_{ij} \preceq e''_{ij}$ and $e'_{ij} \preceq e''_{ij}$; the assignments $e_{ij}$ and $e'_{ij}$ have no ordering. For studying the monotonicity properties involved, we now have to compare the probability distributions over the output variable $C$ that are computed from the Bayesian network under study, given the various assignments to $X$ and $Y$. If for the pair of distributions $\Pr(C \mid e_{ij})$ and $\Pr(C \mid e''_{ij})$, we have that $\Pr(C \mid e''_{ij})$ is stochastically dominant over $\Pr(C \mid e_{ij})$, then the network exhibits the associated property of isotonicity in $X$. If for the distributions given $e'_{ij}$ and $e''_{ij}$ we have that $\Pr(C \mid e'_{ij})$ is stochastically dominant over $\Pr(C \mid e''_{ij})$, then the network shows the associated property of antitonicity in $Y$. The network thus reveals both properties if

$$\Pr(C \mid e_{ij}) \leq \Pr(C \mid e''_{ij}) \leq \Pr(C \mid e'_{ij})$$

Verifying whether or not the network is isotone in $X$ and antitone in $Y$ now amounts to studying the above inequalities for all values $x^i$ and $y^j$ of the two variables.

### 3.3 Deriving offending contexts

Upon using the assignment lattice as described above, some properties of monotonicity may not show in the Bayesian network under study. We then say that these properties are violated. More formally, we say that a pair of joint value assignments $e, e'$ with $e \preceq e'$ *violates* the property of isotonicity if we have $\Pr(C \mid e) > \Pr(C \mid e')$ for their associated probability distributions; violation of the property of antitonicity is defined similarly. We observe that for any violating pair $e, e'$ whose elements are directly linked in the assignment lattice, there is a *unique* variable $E_i$ in the set of observable variables $E$ to which $e$ and $e'$ assign a different value. Let $e_i^j$ be the value of this variable within the assignment $e$ and let $e_i^k$, $k \neq j$, be its value in $e'$. The violation now is denoted by $(e, e' \mid E_i^{j \to k})$. We say that the violation has a change of the value of $E_i$ from $e_i^j$ to $e_i^k$ for its *origin*, written $E_i^{j \to k}$. The value assignment $e^-$ to $E \setminus \{E_i\}$ that is shared by $e$ and $e'$, is called the *context* of the violation.

In general, using the assignment lattice for a Bayesian network may result in a set $\mathcal{V}$ of violations of the properties of monotonicity. Some of the identified violations may show considerable regularity, in the sense that if a violation originates from a change of value in a particular context, then this same change causes a violation in all higher-ordered contexts as well. Such violations will be termed *structural*. Other violations from the set $\mathcal{V}$ do not have such a regular structure and may be considered *incidental*. We distinguish between the two types of violation to allow a compact representation of the entire set of identified violations.

We consider the subset $\mathcal{V}(E_i^{j \to k}) \subseteq \mathcal{V}$ of all violations of the properties of monotonicity that originate from a change of the value of the variable $E_i$ from $e_i^j$ to $e_i^k$. The context $e^-$ of such a violation now is called a *structural context of offence* if we have that there is a violation $(e, e' \mid E_i^{j \to k}) \in \mathcal{V}(E_i^{j \to k})$ for all value assignments $e$ to $E$ that include this context $e^-$ or a higher-ordered one. If there is no lower-ordered structural context of offence for the same change of value $E_i^{j \to k}$, we say that the context of offence is *structurally minimal*. A structurally minimal context of offence thus characterises a set of re-

lated violations. A structurally minimal context $e^-$ for the violations originating from $E_i^{j\to k}$ more specifically defines a sublattice of the assignment lattice in which each element includes $e_i^j$ and in which for each element $e$ there exists a violation $(e, e' \mid E_i^{j\to k})$ in the set $\mathcal{V}(E_i^{j\to k})$; the context $e^-$ is the bottom of this sublattice.

In addition to the violations that are covered by structurally minimal contexts of offence, the set of all identified violations may include elements that are not structurally related. A context $e^-$ is said to be an *incidental context of offence* if it is not a structural context of offence. We note that any set of violations identified from a Bayesian network is now characterised by a set of structurally minimal contexts of offence and a set of incidental contexts.

As an example, we consider the lattice from Figure 1 for the two ternary observable variables $X$ and $Y$. Suppose that upon verifying isotonicity of the Bayesian network under study, the three violations $(x^1y^2, x^2y^2 \mid X^{1\to 2}), (x^1y^3, x^2y^3 \mid X^{1\to 2})$ and $(x^2y^2, x^3y^2 \mid X^{2\to 3})$ are identified. The first two of these violations have the same origin: the two violations arise if the value of the variable $X$ is changed from $x^1$ to $x^2$. These violations are characterised by the structurally minimal context of offence $y^2$: in both the context $y^2$ and the higher-ordered context $y^3$, changing the value of $X$ from $x^1$ to $x^2$ gives rise to a violation. The third violation mentioned above is not yet covered by the identified minimal context since it has a different origin. This violation originates from a change of the value of the variable $X$ from $x^2$ to $x^3$ and again has $y^2$ for its offending context. The context $y^2$ now is merely incidental since no violation has been identified for the pair of assignments $x^2y^3$ and $x^3y^3$ with include the higher-ordered context $y^3$.

To conclude, we would like to note that for a given set of violations, the structurally minimal contexts of offence are readily established by traversing the assignment lattice under study. Starting at the bottom of the lattice, the procedure finds an element $L(e)$ such that $e$ is the lowest-ordered joint value assignment occurring in a violation $(e, e' \mid E_i^{j\to k}) \in \mathcal{V}$. The procedure subsequently isolates from the assignment lattice the sublattice with the element $L(e)$, with $e \equiv e_i^j e^-$, for its bottom; the sublattice further includes all elements $L(e_i^j e^+)$ with $e_i^j e^- \preceq e_i^j e^+$. Now, if each element of the sublattice occurs in a violation with $E_i^{j\to k}$ for its origin, then the context $e^-$ is a structurally minimal context of offence; otherwise it is incidental. The procedure is iteratively repeated for all yet uncovered violations.

### 3.4 Complexity considerations

The runtime complexity of our method for studying monotonicity of a Bayesian network is determined by the size of the assignment lattice used. We observe that this lattice encodes an exponential number of value assignments to the set of observable variables. Constructing the lattice and computing the probability distributions to be associated with its elements, therefore, takes exponential time. Moreover, the dominance properties of an exponential number of pairs of probability distributions have to be compared. For $n$ binary observable variables, for example, already

$$\sum_{i=0}^{n} \binom{n}{i} \cdot (n-i) > 2^n$$

comparisons are required. From these considerations we have that our method has a very high runtime complexity. Although its requirements can be reduced to at least some extent by exploiting the independences modelled in a Bayesian network, for larger networks including a large number of observable variables studying monotonicity inevitably becomes infeasible. The unfavourable computational complexity of the problem, however, is likely to forestall the design of essentially more efficient methods.

In view of the high runtime complexity involved, we propose to use our method for studying properties of *partial monotonicity* only. The concept of partial monotonicity applies to a subset $X$ of the observable variables of a network. An assignment lattice is constructed for these variables as described above. The probability distributions associated with the elements

of the lattice are conditioned on a *fixed* joint value assignment $e^-$ to the observable variables $E^- = E \setminus X$ that are not included in the study. With each element $L(x)$ of the lattice thus is associated the conditional probability distribution $\Pr(C \mid xe^-)$. The lattice now provides for studying the monotonicity properties of the network for the set $X$ given $e^-$. The assignment $e^-$ then is termed the *background assignment* for studying the partial monotonicity.

We would like to note that partial monotonicity of a network given a particular background assignment $e^-$ does not guarantee partial monotonicity given any other background assignment. Also, violations of the properties of monotonicity that are identified given a particular background assignment may not occur given another such assignment. The background assignment against which partial monotonicity is to be studied should therefore be chosen with care and be based upon considerations from the domain of application.

## 4 An Example Application

We applied our method for studying monotonicity to a real Bayesian network in veterinary science. We briefly introduce the network and describe the violations that we identified.

### 4.1 A network for classical swine fever

In close collaboration with two experts from the Central Institute of Animal Disease Control in the Netherlands, we are developing a Bayesian network for the detection of classical swine fever. Classical swine fever is an infectious disease of pigs, which has serious socio-economical consequences upon an outbreak. As the disease has a potential for rapid spread, it is imperative that its occurrence is detected in the early stages. The network under construction is aimed at supporting veterinary practitioners in the diagnosis of the disease when visiting pig farms with disease problems of unknown cause. Our network currently includes 42 variables for which over 2400 parameter probabilities have been assessed. The variables model the pathogenesis of the disease as well as the clinical signs observed in individual pigs. 24 of the total of 42 variables are observable. Figure 2 depicts the graphical structure of the current network.

### 4.2 Constructing the lattice

During the elicitation interviews, our veterinary experts produced various statements that suggested monotonicity. They indicated, for example, that the output variable *CSF Viraemia* should behave isotonically in terms of the five variables *Diarrhoea*, *Ataxia*, *Fever*, *Malaise*, and *Skin haemorrhages*. We focus on these observable variables to illustrate the application of our method for studying partial monotonicity.

From the five observable variables under study, we constructed an assignment lattice as described in the previous section. Since all variables involved were binary, adopting one of the values *true* and *false*, we used a slightly more concise encoding of their joint value assignments. For each assignment $e$ to the set of observable variables $E$, we let $L(e)$ be the subset of $E$ such that $E_i \in L(e)$ if and only if $E_i = \mathit{true}$ occurs in $e$. The elements of the resulting lattice thus are subsets of $E$; the bottom of the lattice is the empty set and the top equals the entire set $E$. The resulting lattice for the five variables under study is shown in Figure 3. It includes $2^5 = 32$ elements to capture all possible joint value assignments to the variables and further includes 80 direct set-inclusion statements.

Before the lattice could be enhanced with conditional probabilities of the presence of a viraemia of classical swine fever, we had to decide upon a background assignment for the other 19 observable variables against which the properties of partial monotonicity would be verified. We decided to take for this purpose the value assignment in which all other observable variables of the network had adopted the value *false*. We chose this particular assignment since the various clinical signs have a rather small probability of occurrence and it is highly unlikely to find a large number of these signs in a single live pig. Given this background assignment, we computed the various conditional probabilities to be associated with the elements of the assignment lattice.

For each pair of directly linked elements from

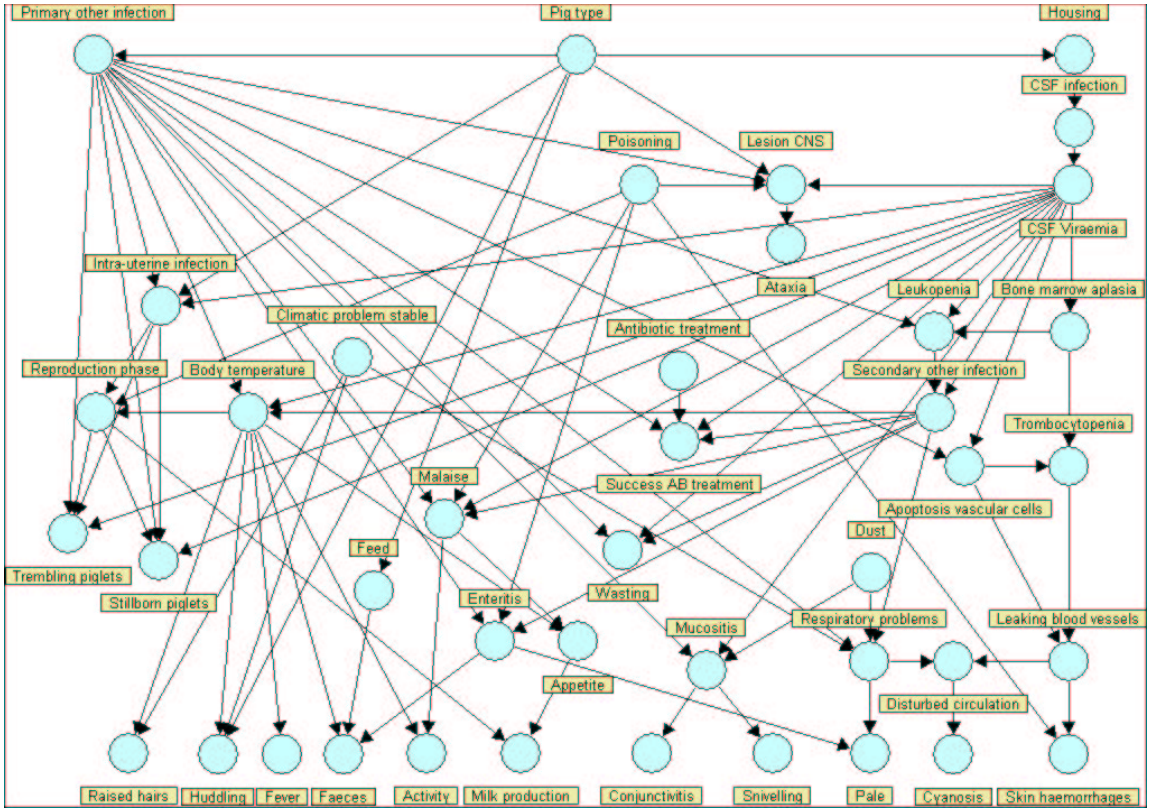



Figure 2: The graphical structure of our Bayesian network for classical swine fever in pigs.

the assignment lattice, we compared the conditional probabilities of a viraemia of classical swine fever. We found four violations of the properties of partial isotonicity given the selected background assignment; these violations are shown by dashed lines in Figure 3. The four violations all originated from adding the clinical sign of diarrhoea to the combination of findings of ataxia and malaise. The four violations thus were all covered by a single structurally minimal context of offence.

We presented the pairs of violating assignments to two veterinarians. Being confronted with the four violations, they independently and with conviction indicated that the probability of a viraemia of classical swine fever should increase upon finding the additional sign of diarrhoea. Both veterinarians mentioned that the combination of ataxia and diarrhoea especially pointed to classical swine fever; within the scope of the Bayesian network, they could not think of another disease that would be more likely to give rise to this combination of signs. They thus indicated that the network should indeed have been isotone in the five variables under study given the absence of any other signs. The four identified violations thus were indicative of modelling inadequacies in our current network.

## 5 Conclusions

In this paper, we have presented a method for studying monotonicity of Bayesian networks. In view of the unfavourable complexity of the problem in general, the method focuses on just a subset of the observable variables of a network and builds upon a lattice of all joint value assignments to these variables. The lattice is enhanced with information about the effects of these assignments on the probability distribution over the network's main output variable. The enhanced lattice then is used for identifying all violations of the properties of monotonicity and for constructing minimal offending contexts for further consideration.

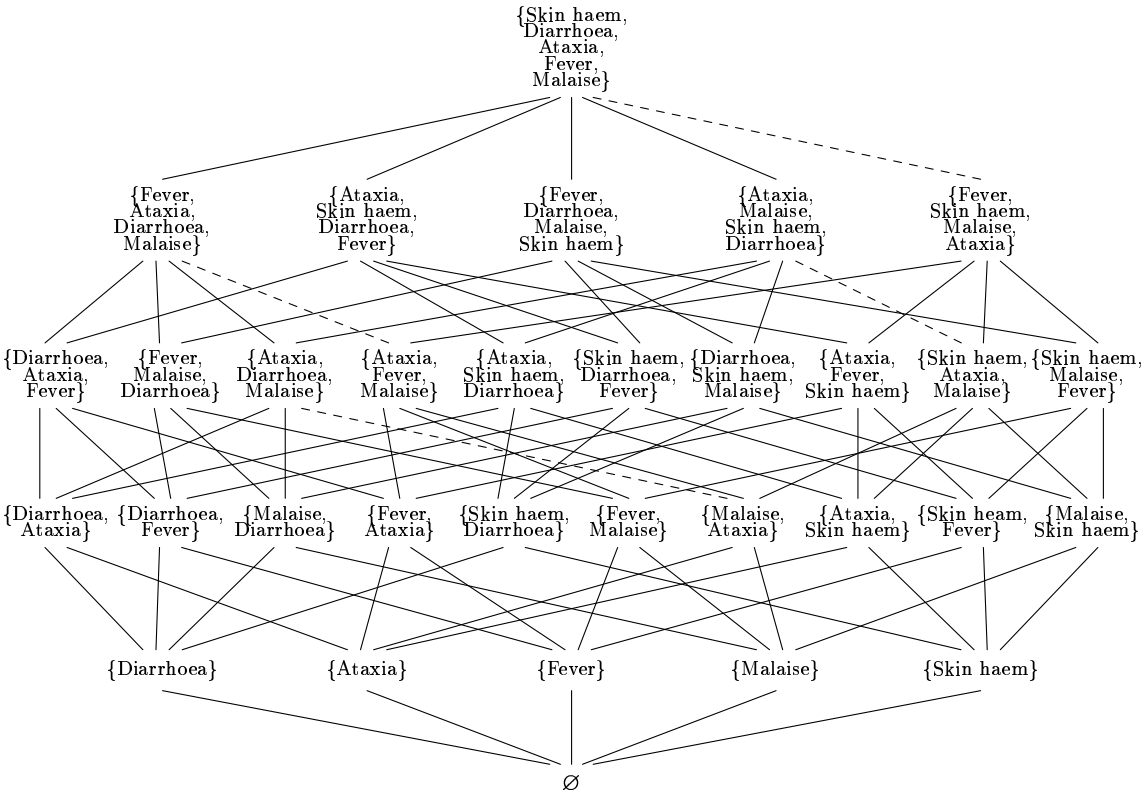



Figure 3: An assignment lattice for our Bayesian network for classical swine fever; the violations of the properties of partial monotonicity are indicated by dashed lines.

We would like to note that, as a supplement to our method for studying monotonicity, we designed a special-purpose elicitation technique that allows for discussing with domain experts whether or not the identified violations indeed can be construed as violations of commonly acknowledged patterns of monotonicity (Van der Gaag *et al.*, 2006). This technique has been designed specifically so as to ask little time and little cognitive effort from the experts in the verification of the identified violations.

The results that we obtained from applying our method for studying monotonicity to a real network in veterinary science, indicate that it presents a useful method for studying reasoning patterns in Bayesian networks. The next step now is to extend our method by techniques that exploit the constructed minimal offending contexts for identifying the modelling inadequacies in a network that cause the various violations of monotonicity.

## References

J.O. Berger. 1980. *Statistical Decision Theory and Bayesian Analysis*, 2nd ed., Springer Verlag.

G. Grätzer. 1971. *Lattice Theory: First Concepts and Distributive Lattices*, W.H. Freeman.

C.-L. Liu and M.P. Wellman. 1998. Incremental tradeoff resolution in quantitative probabilistic networks. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pages 338 – 345.

L.C. van der Gaag, H.L. Bodlaender and A. Feelders. 2004. Monotonicity in Bayesian networks. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, AUAI, pages 569 – 576.

L.C. van der Gaag, P.L. Geenen and H.J.M. Tabachneck-Schijf. 2006. Verifying monotonicity of Bayesian networks with domain experts. In *Proceedings of the 4th Bayesian Modelling Applications Workshop*, pages 9 – 15.

M.P. Wellman. 1990. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44: 257 – 303.

# Multi-dimensional Bayesian Network Classifiers

Linda C. van der Gaag and Peter R. de Waal
Department of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508TB Utrecht, The Netherlands
{linda,waal}@cs.uu.nl

**Abstract**

We introduce the family of multi-dimensional Bayesian network classifiers. These classifiers include one or more class variables and multiple feature variables, which need not be modelled as being dependent on every class variable. Our family of multi-dimensional classifiers includes as special cases the well-known naive Bayesian and tree-augmented classifiers, yet offers better modelling capabilities than families of models with a single class variable. We describe the learning problem for a subfamily of multi-dimensional classifiers and show that the complexity of the solution algorithm is polynomial in the number of variables involved. We further present some preliminary experimental results to illustrate the benefits of the multi-dimensionality of our classifiers.

## 1 Introduction

Bayesian network classifiers have gained considerable popularity for solving classification problems where an instance described by a number of features has to be classified in one of several distinct classes. The success of especially naive Bayesian classifiers and the more expressive tree-augmented network classifiers is readily explained from their ease of construction and their generally good classification performance.

Many application domains, however, include classification problems where an instance has to be assigned to a most likely combination of classes. Since the number of class variables in a Bayesian network classifier is restricted to one, such problems cannot be modelled straightforwardly. One approach is to construct a compound class variable that models all possible combinations of classes. This class variable then easily ends up with an inhibitively large number of values. Also, the structure of the problem is not properly reflected in the resulting model. Another approach is to develop multiple classifiers, one for each original class. Multiple classifiers, however, cannot capture interactions among the various classes and may thus also not properly reflect the problem. Moreover, if the various classifiers indicate multiple classes, then the implied combination may not be the most likely explanation of the observed features.

In this paper we introduce the concept of multi-dimensionality in Bayesian network classifiers to provide for accurately modelling problems where instances are assigned to multiple classes. A multi-dimensional Bayesian network classifier includes one or more class variables and one or more feature variables. It models the relationships between the variables by acyclic directed graphs over the class variables and over the feature variables separately, and further connects the two sets of variables by a bi-partite directed graph; an example multi-dimensional classifier is depicted in Figure 1. As for one-dimensional Bayesian network classifiers, we distinguish between different types of multi-dimensional classifier by imposing restrictions on their graphical structure. Fully tree-augmented multi-dimensional classifiers, for example, have directed trees over their class variables as well as over their feature variables.

For the family of fully tree-augmented multi-dimensional classifiers, we study the learning problem, that is, the problem of finding a classifier that best fits a set of available data. We show that, given a fixed selection of feature vari-

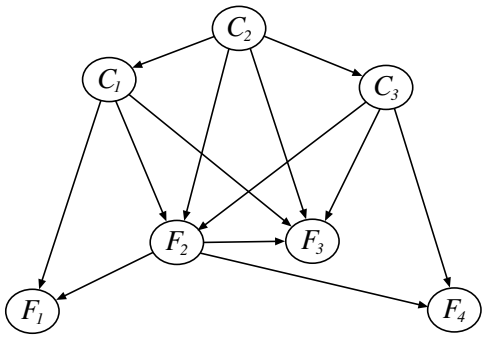



Figure 1: An example multi-dimensional Bayesian network classifier with class variables $C_i$ and feature variables $F_j$.

ables per class variable, the learning problem can be decomposed into optimisation problems for the set of class variables and for the set of feature variables separately, which can both be solved in polynomial time. We further argue that, although our learning algorithm assumes a fixed bi-partite graph between the class and feature variables, it is easily combined with existing approaches to feature subset selection.

The numerical results that we obtained from preliminary experiments with our learning algorithm clearly illustrate the benefits of multi-dimensionality of Bayesian network classifiers. Especially on smaller data sets, the constructed multi-dimensional classifiers provided higher accuracy than their one-dimensional counterparts. In combination with feature selection, moreover, our algorithm resulted in sparser classifiers with considerably fewer parameters and, hence, with smaller variance.

The paper is organised as follows. In Section 2, we review Bayesian network classifiers in general. In Section 3, we define our family of multi-dimensional classifiers. In Section 4, we address the learning problem for fully tree-augmented multi-dimensional classifiers and present a polynomial-time algorithm for solving it. In Section 5, we briefly address feature selection for our multi-dimensional classifiers. We report some preliminary results from an application in the biomedical domain in Section 6. The paper is rounded off with our concluding observations in Section 7.

## 2 Preliminaries

Before reviewing naive Bayesian and tree-augmented network classifiers, we introduce our notational conventions. We consider Bayesian networks over a finite set $V = \{X_1, \ldots, X_k\}$, $k \geq 1$, of discrete random variables, where each $X_i$ takes a value in a finite set $Val(X_i)$. For a subset of variables $Y \subseteq V$ we use $Val(Y) = \times_{X_i \in Y} Val(X_i)$ to denote the set of joint value assignments to $Y$. A Bayesian network now is a pair $B = \langle G, \Theta \rangle$, where $G$ is an acyclic directed graph whose vertices correspond to the random variables $V$ and $\Theta$ is a set of parameter probabilities; the set $\Theta$ includes a parameter $\theta_{x_i|\Pi x_i}$ for each value $x_i \in Val(X_i)$ and each value assignment $\Pi x_i \in Val(\Pi X_i)$ to the set $\Pi X_i$ of parents of $X_i$ in $G$. The network $B$ now defines a joint probability distribution $P_B$ over $V$ that is factorised according to

$$P_B(X_1, \ldots, X_k) = \prod_{i=1}^{k} \theta_{X_i|\Pi X_i}.$$

Bayesian network classifiers are Bayesian networks of restricted topology that are tailored to solving classification problems where instances described by a number of features have to be classified in one of several distinct predefined classes (Friedman *et al.*, 1997). The set of variables $V$ of a Bayesian network classifier is partitioned into a set $V_F = \{F_1, \ldots, F_m\}$, $m \geq 1$, of *feature variables* and a singleton set $V_C = \{C\}$ with the *class variable*. A *naive Bayesian classifier* has a directed tree for its graph $G$, in which the class variable $C$ is the unique root and each feature variable $F_i$ has $C$ for its only parent. A *tree-augmented network* (*TAN*) *classifier* has for its graph $G$ a directed acyclic graph in which the class variable $C$ is the unique root and each feature variable $F_i$ has $C$ and at most one other feature variable for its parents; the subgraph induced by the set $V_F$, moreover, is a directed tree, termed the *feature tree* of the classifier.

The general problem of learning a Bayesian network classifier from a given set of data samples $D = \{u_1, \ldots, u_N\}$, $N \geq 1$, is to find from among the family of network classifiers one that best matches the available data. As a measure

of how well a model describes the data, often the model's log-likelihood given the data is used; for a Bayesian network $B$ and a data set $D$, the log-likelihood of $B$ given $D$ is defined as

$$LL(B \mid D) = \sum_{i=1}^{N} \log \Big( P_B(u_i) \Big).$$

For the family of Bayesian network classifiers in general, the learning problem is intractable. For various subfamilies of classifiers, however, the problem is solvable in polynomial time. Examples include the naive Bayesian and TAN classifiers reviewed above and the subfamily of Bayesian network classifiers of which the subgraph induced by the feature variables constitutes a forest (Lucas, 2004). For other subfamilies of Bayesian network classifiers, researchers have presented heuristic algorithms which provide good, though non-optimal, solutions (Sahami, 1996; Keogh and Pazzani, 1999). We would like to note that these results all relate to learning a classifier for a fixed set of relevant feature variables. To the best of our knowledge, the selection of an optimal set of feature variables has not been solved as yet.

Since its graph has a fixed topology, the problem of learning a naive Bayesian classifier amounts to just establishing maximum-likelihood estimates from the available data for its parameters. For a fixed graphical structure in general, the maximum-likelihood estimators of the parameter probabilities are given by

$$\hat{\theta}_{x_i|\Pi x_i} = \hat{P}_D(x_i \mid \Pi x_i),$$

where $\hat{P}_D$ denotes the empirical distribution defined by the frequencies of occurrence in the data. The problem of learning a TAN classifier amounts to first determining a graphical structure of maximum likelihood given the available data and then establishing estimates for its parameters. For constructing a maximum-likelihood feature tree, a polynomial-time algorithm is available from Friedman *et al.* (1997).

To conclude, we would like to note that a Bayesian network classifier computes for each instance a conditional probability distribution over the class variable. For classification purposes, it further is associated with a function $\mathcal{C}\colon \mathit{Val}(V_F) \rightarrow \mathit{Val}(C)$ which typically builds upon the *winner-takes-all rule*. Using this rule, a classifier $B$ outputs for each value assignment $f \in \mathit{Val}(V_F)$, a class value $c$ such that $P_B(c \mid f) \geq P_B(c' \mid f)$ for all $c' \in \mathit{Val}(C)$, breaking ties at random.

## 3 Multi-dimensional Classifiers

Bayesian network classifiers as reviewed above, include a single class variable and as such are one-dimensional. We now introduce the concept of multi-dimensionality in Bayesian network classifiers by defining a family of models that may include multiple class variables.

A *multi-dimensional Bayesian network classifier* is a Bayesian network of which the graph $G = \langle V, A \rangle$ has a restricted topology. The set $V$ of random variables is partitioned into the sets $V_C = \{C_1, \ldots, C_n\}$, $n \geq 1$, of class variables and the set $V_F = \{F_1, \ldots, F_m\}$, $m \geq 1$, of feature variables. The set of arcs $A$ of the graph is partitioned into the three sets $A_C$, $A_F$ and $A_{CF}$ having the following properties:

- for each $F_i \in V_F$ there is a $C_j \in V_C$ with $(C_j, F_i) \in A_{CF}$ and for each $C_i \in V_C$ there is an $F_j \in V_F$ with $(C_i, F_j) \in A_{CF}$;
- the subgraph of $G$ that is induced by $V_C$ equals $G_C = \langle V_C, A_C \rangle$;
- the subgraph of $G$ that is induced by $V_F$ equals $G_F = \langle V_F, A_F \rangle$.

The subgraph $G_C$ is a graphical structure over the class variables and is called the classifier's *class subgraph*; the subgraph $G_F$ is called its *feature subgraph*. The subgraph $G_{CF} = \langle V, A_{CF} \rangle$ is a bi-partite graph that relates the various feature variables to the class variables; this subgraph is called the *feature selection subgraph* of the classifier and its set of arcs is termed the classifier's *feature selection arc set*. For any variable $X$ in a multi-dimensional classifier, we now use $\Pi_C X$ to denote the class parents of $X$ in $G$, that is, $\Pi_C X = \Pi X \cap V_C$. We further use $\Pi_F X$ to denote the feature parents of $X$ in $G$. Note that for any class variable $C_i$ we thus have that $\Pi_F C_i = \varnothing$ and $\Pi_C C_i = \Pi C_i$.

Within the family of multi-dimensional Bayesian network classifiers various different types of classifier are distinguished based upon their graphical structures. An example is the *fully naive multi-dimensional classifier* in which both the class subgraph and the feature subgraph have empty arc sets. Note that this subfamily of bi-partite classifiers includes the one-dimensional naive Bayesian classifier as a special case. Reversely, any such bi-partite classifier has an equivalent naive Bayesian classifier with a single compound class variable. Another type of multi-dimensional classifier is the subfamily of classifiers in which both the class subgraph and the feature subgraph are directed trees. In the remainder of the paper, we will focus on this subfamily of *fully tree-augmented multi-dimensional classifiers*.

A multi-dimensional classifier in essence is used to find a joint value assignment of highest posterior probability to its set of class variables. Finding such an assignment given values for all feature variables involved, is equivalent to solving the MPA problem. This problem is known to be NP-hard in general, yet can be solved in polynomial time for networks of bounded treewidth (Bodlaender *et al.*, 2002). In the presence of unobserved feature variables, the problem of finding assignments of highest posterior probability remains intractable even for these restricted networks (Park, 2002). In view of the unfavourable computational complexity involved, we note that the practicability of multi-dimensional classifiers is limited to models with restricted class subgraphs.

## 4 Learning Fully Tree-augmented Multi-dimensional Classifiers

In this section we define the problem of learning a fully tree-augmented multi-dimensional classifier and show that this problem can be decomposed into two separate optimisation problems which can both be solved in polynomial time.

### 4.1 The learning problem

Before defining the problem of learning a fully tree-augmented multi-dimensional classifier from data, we recall that the related problem for Bayesian network classifiers has been studied for a fixed set of relevant feature variables. Following a similar approach, we now formulate our learning problem to pertain to a subfamily of classifiers for which the feature selection subgraph is fixed. We will return to the issue of feature subset selection in Section 5.

We begin by defining the subfamily of fully tree-augmented classifiers with a fixed selection of feature variables per class variable. These classifiers are considered *admissible* for the learning problem. We let the set of random variables $V$ be partitioned into $V_C$ and $V_F$; we further take a set of arcs $A$ to be partitioned into $A_C$, $A_F$ and $A_{CF}$ as before. We now consider a subset $\underline{A}_{CF}$ of $V_C \times V_F$ such that $\langle V, \underline{A}_{CF} \rangle$ is a feature selection subgraph. A fully tree-augmented multi-dimensional classifier now is admissible for $\underline{A}_{CF}$ if we have for its set of arcs $A_{CF}$ that $A_{CF} = \underline{A}_{CF}$. The set of all admissible classifiers for $\underline{A}_{CF}$ is denoted as $\mathcal{B}_{\underline{A}_{CF}}$.

The learning problem now is to find from among the set of admissible classifiers one that best fits the available data. As a measure of how well a model describes the data, we use its log-likelihood given the data. More formally, the learning problem for fully tree-augmented multi-dimensional classifiers with a fixed feature selection arc set $\underline{A}_{CF}$ then is to find a classifier $B$ in $\mathcal{B}_{\underline{A}_{CF}}$ that maximises $LL(B \mid D)$.

### 4.2 Solving the learning problem

In this section we show that the learning problem for fully tree-augmented multi-dimensional classifiers can be solved in polynomial time.

We consider a fully tree-augmented classifier $B$ with the class variables $V_C$ and the feature variables $V_F$ that is admissible for the feature selection arc set $\underline{A}_{CF}$. Building upon a result from Friedman *et al.* (1997), we have that the log-likelihood of $B$ given a data set $D$ can be written as

$$LL(B \mid D) =$$

$$= -N \cdot \sum_{i=1}^{n} H_{\hat{P}_D}(C_i | \Pi C_i) + N \cdot \sum_{j=1}^{m} H_{\hat{P}_D}(F_j | \Pi F_j)$$

$$= N \cdot \sum_{i=1}^{n} I_{\hat{P}_D}(C_i; \Pi C_i) - N \cdot \sum_{i=1}^{n} H_{\hat{P}_D}(C_i)$$
$$+ N \cdot \sum_{j=1}^{m} I_{\hat{P}_D}(F_j; \Pi F_j) - N \cdot \sum_{j=1}^{m} H_{\hat{P}_D}(F_j),$$

where $\hat{P}_D$ is the empirical distribution from $D$, $H_P(X) = -\sum_x P(x) \cdot \log P(x)$ is the entropy of a random variable $X$ with distribution $P$, $H_P(X \mid Y) = -\sum_{x,y} P(x,y) \cdot \log P(x \mid y)$ denotes the conditional entropy of $X$ given $Y$, and

$$I_P(X;Y) = \sum_{x,y} P(x,y) \cdot \log \frac{P(x,y)}{P(x) \cdot P(y)}$$

denotes the mutual information of $X$ and $Y$.

The two entropy terms $H_{\hat{P}_D}(C_i)$ and $H_{\hat{P}_D}(F_j)$ in the above expression for $LL(B \mid D)$ concern marginal distributions established from the available data. These terms therefore depend only on the empirical distribution and not on the graphical structure of the classifier. This observation implies that an admissible classifier that maximises the log-likelihood given the data is a classifier that maximises the sum of its two mutual-information terms.

We consider the mutual-information term $I_{\hat{P}_D}(F_j; \Pi F_j)$ in some more detail. We note that the set of parents $\Pi F_j$ of any feature variable $F_j$ is partitioned into the set $\Pi_C F_j$ of class parents and the set $\Pi_F F_j$ of feature parents. Using the chain rule for mutual information (Cover and Thomas, 1991), the term $I_{\hat{P}_D}(F_j; \Pi F_j)$ can now be written as

$$\sum_{j=1}^{m} \Big( I_{\hat{P}_D}(F_j; \Pi_C F_j) + I_{\hat{P}_D}(F_j; \Pi_F F_j \mid \Pi_C F_j) \Big),$$

where $I_P(X;Y \mid Z)$ is the conditional mutual information of $X$ and $Y$ given $Z$ with

$$I_P(X;Y \mid Z) =$$
$$= \sum_{x,y,z} P(x,y,z) \cdot \log \frac{P(x,y \mid z)}{P(x \mid z) \cdot P(y \mid z)}.$$

Since the feature selection arc set $\underline{A}_{CF}$ is fixed, the set $\Pi_C F_j$ of class parents of the feature variable $F_j$ is the same for every admissible classifier. We conclude that the mutual-information term $I_{\hat{P}_D}(F_j; \Pi_C F_j)$ is the same for all models in the set of admissible classifiers $\mathcal{B}_{\underline{A}_{CF}}$.

To summarise the above considerations, we have that a classifier that solves the learning problem for fully tree-augmented multi-dimensional classifiers with the fixed feature selection arc set $\underline{A}_{CF}$, is a classifier from $\mathcal{B}_{\underline{A}_{CF}}$ that maximises

$$\sum_{i=1}^{n} I_{\hat{P}_D}(C_i; \Pi C_i) + \sum_{j=1}^{m} I_{\hat{P}_D}(F_j; \Pi_F F_j \mid \Pi_C F_j).$$

We now proceed by showing that the learning problem can be decomposed into two separate optimisation problems which can both be solved in polynomial time. To this end, we first consider the mutual-information term pertaining to the class variables. We observe that, since class variables only have class parents, this term depends on the set of arcs $A_C$ of the class subgraph only. With respect to the conditional mutual-information term above, we observe that this term depends on the feature selection arc set $\underline{A}_{CF}$, which is fixed, and on $A_F$, but not on the set $A_C$. These considerations imply that the two terms are independent and can be maximised separately.

The mutual-information term pertaining to the class variables can be maximised by using the procedure from Chow and Liu (1968) for constructing maximum-likelihood trees:

1. Construct a complete undirected graph over the set of class variables $V_C$.
2. Assign a weight $I_{\hat{P}_D}(C_i, C_j)$ to each edge between $C_i$ and $C_j$, $i \neq j$.
3. Build a maximum-weighted spanning tree, for example using Kruskal's algorithm (1956).
4. Transform the undirected tree into a directed one, by choosing an arbitrary variable for its root and setting all arc directions from the root outward.

For the conditional mutual-information term pertaining to the feature variables, we observe that it is maximised by finding a maximum-likelihood directed spanning tree over the feature variables. Such a tree is constructed by the following procedure:

1. Construct a complete directed graph over the set of variables $V_F$.
2. Assign a weight $I_{\hat{P}_D}(F_i; F_j \mid \Pi_C F_j)$ to each arc from $F_i$ to $F_j$, $i \neq j$.
3. Build a maximum-weighted directed spanning tree, for example using the algorithm of Chu and Liu (1968) or Edmonds' algorithm (1967).

We would like to note that for maximising the conditional mutual-information term for the feature variables, we have to construct a directed spanning tree, while for maximising the mutual-information term for the class variables we compute an undirected one. The need for this difference arises from the observation that $I_{\hat{P}_D}(C_i; C_j) = I_{\hat{P}_D}(C_j; C_i)$ for the class variables while $I_{\hat{P}_D}(F_i; F_j \mid \Pi_C F_j) \neq I_{\hat{P}_D}(F_j; F_i \mid \Pi_C F_i)$ for the feature variables.

Our algorithm for solving the learning problem for fully tree-augmented multi-dimensional classifiers with a fixed feature selection subgraph is composed of the two procedures described above. Solving the problem thus amounts to computing an undirected maximum-weighted spanning tree over the class variables and a directed maximum-weighted spanning tree over the feature variables. The computation of the weights for the undirected tree has a computational complexity of $O(n^2 N)$, while the construction of the tree itself requires $O(n^2 \log n)$ time (Kruskal, 1956). The computation of the weights for the directed tree has a complexity of $O(m^2 N)$, while the construction of the tree itself takes $O(m^3)$ time. Since a typical data set satisfies $N > \log n$ and $N > m$, we have that the overall complexity of our algorithm is polynomial in the number of variables involved.

We conclude this section by observing that the learning problem can also be formulated for classifiers in which the set $A_C$ or the set $A_F$ is empty. Our algorithm is readily adapted to these problems. With $A_C = \varnothing$, only the conditional mutual-information term for the feature variables has to be maximised using the second procedure above. With $A_F = \varnothing$, only the mutual-information term for the class variables has to be maximised using the first procedure.

## 5 Feature Subset Selection

In the previous section, we have addressed the learning problem for fully tree-augmented multi-dimensional classifiers with a fixed feature selection subgraph. We now briefly discuss feature subset selection for our classifiers.

It is well known that, if more or less redundant features are included in a data set, these features may bias the classifier that is learned from the data, which in turn may result in a relatively poor classification accuracy. By constructing the classifier over just a subset of the feature variables, a less complex classifier is yielded that tends to have a better performance (Langley *et al.*, 1992). Finding a minimum subset of features such that the selective classifier constructed over this subset has highest accuracy is known as the feature subset selection problem. The feature selection problem unfortunately is known to be NP-hard in general (Tsamardinos and Aliferis, 2003).

For Bayesian network classifiers, different heuristic approaches to feature subset selection have been proposed. One of these is the *wrapper approach* (Kohavi and John, 1997), in which the selection of feature variables is merged with the learning algorithm. We now argue that the same approach can be used for our multi-dimensional Bayesian network classifiers. The resulting procedure is as follows:

1. Choose the empty feature selection subgraph for the initial current subgraph.
2. From the current subgraph generate all possible feature selection subgraphs that are obtained by adding an arc from a class variable to a feature variable.
3. For each generated feature selection subgraph, compute the accuracy of the best classifier given this subgraph that is learned using the algorithm from the previous section.
4. Select the best generated subgraph, that is, the feature selection subgraph of the classifier of highest accuracy.

5. If the accuracy of the classifier with the selected subgraph is higher than that of the classifier with the current subgraph, then denote the selected subgraph as the current subgraph and go to Step 2. If not, then stop and propose the best classifier for the current subgraph as the overall best.

Starting with an empty graphical structure without any arcs, as in the above procedure, is known as *forward selection*. Alternatively, *backward elimination* can be used, which starts with a full graphical structure from which single arcs are removed in an iterative fashion.

## 6 Experimental Results

In this section we present some preliminary numerical results from our experiments to illustrate the benefits of multi-dimensionality of Bayesian network classifiers.

Since the UCI repository of benchmark data does not include any data sets with multiple class variables, we decided to generate some artificial data sets to test our learning algorithm. These data sets were generated from the oesophageal cancer network (Van der Gaag *et al.*, 2002). This network for the staging of cancer of the oesophagus includes 42 random variables of which 25 are observable feature variables. Three of the network's variables in essence are class variables, which in the current network are summarised in a single output variable. We generated three data sets of 100, 200 and 400 samples, respectively, using logic sampling. From the generated samples we removed the values of all non-observable variables, except for those of the three class variables.

From the three data sets we constructed fully naive and fully tree-augmented multi-dimensional classifiers. For this purpose, we used the learning algorithm described in the previous sections. For comparison purposes, we further learned naive and tree-augmented Bayesian network classifiers with a compound class variable from the data. For all classifiers, we used a forward-selection wrapper approach with the learning algorithm. The accuracies of the various classifiers were calculated using ten-fold cross-validation. For the multi-dimensional classifiers, we defined their accuracy as the proportion of samples that were classified correctly for all class variables involved.

| size of data set: 100 | | |
|---|---|---|
| *classifier type* | *acc.* | *# par.* |
| Compound naive | 0.46 | 695 |
| Multi-dim naive | 0.54 | 136 |
| Compound TAN | 0.35 | 1869 |
| Multi-dim FTAN | 0.41 | 740 |
| size of data set: 200 | | |
| *classifier type* | *acc.* | *# par.* |
| Compound naive | 0.420 | 661 |
| Multi-dim naive | 0.555 | 179 |
| Compound TAN | 0.305 | 3060 |
| Multi-dim FTAN | 0.475 | 1092 |
| size of data set: 400 | | |
| *classifier type* | *acc.* | *# par.* |
| Compound naive | 0.550 | 732 |
| Multi-dim naive | 0.605 | 276 |
| Compound TAN | 0.505 | 4604 |
| Multi-dim FTAN | 0.585 | 386 |

Table 1: Experimental results for different types of classifier on data sets of different size generated from the oesophageal cancer network.

The results from our experiments are summarised in Table 1. The accuracy of the best learned classifier is given in the second column; the third column gives the number of parameter probabilities that were estimated for this classifier. From the table we may conclude that the multi-dimensional classifiers, without exception, outperform their compound counterparts in terms of accuracy. Also the numbers of estimated parameters are considerably smaller for the multi-dimensional classifiers. On average, the learned multi-dimensional classifiers require one-third of the number of parameters of their compound counterparts. The difference is particularly striking for the naive classifiers learned from the 100-sample data set, where the multi-dimensional classifier needs only one-fifth of the number of parameters of the compound one. The smaller numbers of parameters

required constitute a considerable advantage of the multi-dimensional classifiers over their compound counterparts since these parameters typically need to be estimated from relatively small data sets.

Although our preliminary experimental results look promising, we are aware that further experimentation is necessary to substantiate any claims about better performance of our multi-dimensional Bayesian network classifiers.

## 7 Conclusions and Future Research

In this paper we introduced a new family of Bayesian network classifiers that include one or more class variables and multiple feature variables that need not be modelled as being dependent upon every class variable. We formulated the learning problem for this family and presented a solution algorithm that is polynomial in the number of variables involved. Our preliminary experimental results served to illustrate the benefits of the multi-dimensionality of our Bayesian network classifiers.

In the near future we intend to perform a more extensive experimentation study of our learning algorithm, using other data sets and other approaches to feature subset selection. We further wish to investigate other types of model from our family of multi-dimensional classifiers. Possible alternatives include classifiers with $k$-dependence polytrees over their feature variables. Since the number of class variables usually is rather small, we also would like to investigate the feasibility of classifiers with slightly more complex class subgraphs.

## References


H.L. Bodlaender, F. van den Eijkhof and L.C. van der Gaag. (2002). On the complexity of the MPA problem in probabilistic networks. In: *Proceedings of the 15th European Conference on Artificial Intelligence*, IOS Press, pages 675 – 679.

V.K. Chow and C.N. Liu. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14: 462 – 467.

Y.J. Chu and T.H. Liu. (1965). On the shortest arborescence of a directed graph. *Scientia Sinica*, 14: 1396 – 1400.

T.M. Cover and J.A. Thomas. (1991). *Elements of Information Theory*, Wiley.

J. Edmonds. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B: 233 – 240.

N. Friedman, D. Geiger and M. Goldszmidt. (1997). Bayesian network classifiers. *Machine Learning*, 29: 131 – 163.

E.J. Keogh and M.J. Pazzani. (1999). Learning augmented Bayesian classifiers: a comparison of distribution-based and classification-based approaches. In: *Proceedings of the 7th International Workshop on AI and Statistics*, pages 225 – 230.

R. Kohavi and G.H. John. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97: 273 – 324.

J.B. Kruskal. (1956). On the shortest spanning subtree of a graph and the travelling salesman problem. *Proceedings of the American Mathematical Society*, 7: 48 – 50.

P. Langley, W. Iba and K. Thompson. (1992). An analysis of Bayesian classifiers. In: *Proceedings of the 10th Conference on Artificial Intelligence*, pages 223 – 228.

P.J.F. Lucas. (2004). Restricted Bayesian network structure learning. In: *Advances in Bayesian Networks. Studies in Fuzziness and Soft Computing*, vol. 146, Springer-Verlag, pages 217–232.

J. Park. (2002). MAP complexity results and approximation methods. In: *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pages 388 – 396.

M. Sahami. (1996). Learning limited dependence Bayesian classifiers. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, AAAI Press, pages 335 – 338.

I. Tsamardinos and C.F. Aliferis. (2003). Towards principled feature selection: relevance, filters, and wrappers. In: *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics.*

L.C. van der Gaag, S. Renooij, C.L.M. Witteman, B.M.P. Aleman and B.G. Taal. (2002). Probabilities for a probabilistic network: a case study in oesophageal cancer. *Artificial Intelligence in Medicine*, 25: 123 – 148.

# Dependency networks based classifiers: learning models by using independence tests.

José A. Gámez and Juan L. Mateo and José M. Puerta
Computing Systems Department
Intelligent Systems and Data Mining Group – $i^3\mathcal{A}$
University of Castilla-La Mancha
Albacete, 02071, Spain

**Abstract**

In this paper we propose using dependency networks (Heckerman et al., 2000), that is a probabilistic graphical model similar to Bayesian networks, to model classifiers. The main difference between these two models is that in dependency networks cycles are allowed, and this fact has the consequence that the automatic learning process is much easier and can be parallelized. These properties make dependency networks a valuable model especially when it is needed to deal with large databases. Because of these promising characteristics we analyse the usefulness of dependency networks-based Bayesian classifiers. We present an approach that uses independence tests based on chi square distribution, in order to find relationships between predictive variables. We show that this algorithm is as good as some state-of the-art Bayesian classifiers, like TAN and an implementation of the BAN model, and has, in addition, other interesting proprierties like scalability or good quality for visualizing relationships.

## 1 Introduction

Classification is basically the task of assigning a label to an instance formed by a set of attributes. The automatic approach for this task is one of the main parts in machine learning. Thus, the problem consists on learning classifiers from datasets in which each instance has been previously labelled. Many methods have been used for solving this problem based on various representations such as decision trees, neural networks, rule based systems, support vector machines among others. However a probabilistic approach can be used for this purpose, Bayesian classifiers (Duda and Hart, 1973; Friedman et al., 1997). Bayesian classifiers make use of a probabilistic graphical model such as Bayesian networks, that have been a very popular way of representing probabilistic relationships between variables in a domain. Thus a Bayesian classifier takes advantage of probability theory, especially the Bayes rule, in conjunction with a graphical representation for qualitative knowledge. The classification task can be expressed as determining the probability for the class variable knowing the value for all other variables:

$$P(C|X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n).$$

After this computation, the class variable's value with the highest probability is returned as result. Obviously computing this probability distribution for the class variable can be very difficult and inefficient if it is carried out directly, but based on the independencies stated by the selected model this problem can be simplified enormously.

In this paper we present a new algorithm for the automatic learning of Bayesian classifiers from data, but using *dependency network classifiers* instead of Bayesian networks. Dependency networks (DNs) were proposed by Heckerman et al. (2000) as a new probabilistic graphical model similar to BNs, but with a key difference: the graph that encodes the model

structure does not have to be acyclic. As in BNs each node in a DN has a conditional probability distribution given its parents in the graph. Although the presence of cycles can be observed as the capability to represent richer models, the price we have to pay is that usual BNs inference algorithms cannot be applied and Gibbs sampling has to be used in order to recover the joint probability distribution (see (Heckerman et al., 2000)). An initial approach for introducing dependency networks in automatic classification can be found in (Mateo, 2006).

Learning a DN from data is easier than learning a BN, especially because restrictions about introducing cycles are not taken into account. In (Heckerman et al., 2000) was presented a method for learning DNs based on learning the conditional probability distribution for each node independently, by using any classification or regression algorithm to discover the parents of each node. Also a feature selection scheme can be use in conjunction. It is clear that this learning approach produces scalable algorithms, because any learning algorithm with this approach can be easily parallelized just distributing groups of variables between all nodes available in a multi-computer system.

This paper is organized as follows. In section 2 dependency networks are briefly described. In section 3 we review some notes about how classification task is done with probabilistic graphical models. In section 4 we describe the algorithm to learn dependency networks classifiers from data by using independence tests. In section 5 we show the experimental results for this algorithm and compare them with state-of-the-art BN classifiers, like TAN algorithm and an implementation of the BAN model. In section 6 we present our conclusions and some open research lines for the future.

## 2 Preliminaries

A Bayesian network (Jensen, 2001) $\mathcal{B}$ over the domain $\mathbf{X} = \{X_1, X_2, \cdots, X_n\}$, can be defined as a tuple ($\mathbb{G}$,$\mathbf{P}$) where $\mathbb{G}$ is a directed acyclic graph, and $\mathbf{P}$ is a joint probability distribution. Due to the conditional independence constraints encoded in the model, each variable is independent of its non-descendants given its parents. Thus, $P$ can be factorized as follows:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|Pa_i) \qquad (1)$$

A dependency network is similar to a BN, but the former can have directed cycles in its associated graph. In this model, based on its conditional independence constraints, each variable is independent of *all other variables* given its parents. A DN $\mathcal{D}$ can be represented by ($\mathbb{G}$,$\mathbf{P}$) where $\mathbb{G}$ is a directed graph not necessarily acyclic, and as in a BN $\mathbf{P} = \{\forall i, P(X_i|\mathbf{Pa}_i)\}$ is the set of local probability distributions, one for each variable. In (Heckerman et al., 2000) it is shown that inference over DNs, recovering joint probability distribution of $\mathbf{X}$ from the local probability distributions, is done by means of Gibbs sampling instead of the traditional inference algorithms used for BNs due to the potential existence of cycles.

A dependency network is said *consistent* if $P(\mathbf{X})$ can be obtained from $\mathbf{P}$ via its factorization from the graph (equation 1). By this definition, can be shown that exists an equivalence between a consistent DN and a Markov network. This fact suggests an approach for learning DNs from Markov network learned from data. The problem with this approach is that can be computationally inefficient in many cases.

Due to this inconvenient, consistent DNs became very restrictive, so in (Heckerman et al., 2000) the authors defined *general* DNs in which consistency about joint probability distribution is not required. General DNs are interesting for automatic learning due to the fact that each local probability distribution can be learned independently, although this local way of processing can lead to inconsistencies in the joint probability distribution. Furthermore, structural inconsistencies ($X_i \in pa(X_j)$ but not $X_j \in pa(X_i)$) can be found in general DNs. Nonetheless, in (Heckerman et al., 2000) the authors argued that these inconsistencies can disappear, or at least be reduced to the minimum when a reasonably large dataset is used to learn from. In this work we only consider general DNs.

An important concept in probabilistic graphical models is *Markov blanket*. This concept is defined for any variable as the set of variables which made it independent from the rest of variables in the graph. In a BN the Markov blanket is formed by the set of parents, children and parent of the children for every variable. However, in a DN, the Markov blanket for a variable is exactly the parents set. That is the main reason why DN are useful for visualizing relationships among variables, because these relationships are explicitly shown.

## 3 Probabilistic graphical models in classification

As it has been said before, we can use Bayesian models in order to perform a classification task. Within this area, the more classical Bayesian classifier is the one called *naive* (Duda and Hart, 1973; Langley et al., 1992). This model considers all predictive attributes independent given the class variable. It has been shown that this classifier is one of the most effective and competitive in spite of its very restrictive model since it does not allow dependencies between predictive variables. However this model is used in some applications and is employed as initial point for other Bayesian classifiers. In the augmented naive bayes family models, the learning algorithm begins with the structure proposed by naive Bayesian classifier and then try to find dependencies among variables using a variety of strategies. One of this strategies can be looking for relationships between predictive variables represented by a simple structure like a tree as is done in TAN (Tree Augmented Naive Bayes) algorithm (Friedman et al., 1997). Another is to adapt a learning algorithm employed in machine learning with general Bayesian networks. In this case the learning method is modified in order to take into account the relationship between every variable and the class.

Apart from this approach to learn Bayesian classifiers, it must be mentioned another one based on the Markov blanket concept. These methods focus their efforts in searching the set of variables that will be the class variable's Markov blanket. Nonetheless, it has been shown that methods based on extending Naive Bayes classifier, in general, outperform the ones based on the Markov blanket.

When the classifier model has been learned is time to use inference in order to find what class value has to be assigned for the values of a given instance. To accomplish this task the MAP (*Maximum at posterior*) hypothesis is used, which returns as result the value for the class variable that maximize the probability given the values of predictive variables. This is represented by the following expression:

$$\begin{aligned} c_{MAP} &= \arg\max_{c\in\Omega_C} \quad p(c|x_1,\ldots,x_n) \\ &= \arg\max_{c\in\Omega_C} \quad \frac{p(x_1,\ldots,x_n|c)\cdot p(c)}{p(x_1,\ldots,x_n)} \\ &= \arg\max_{c\in\Omega_C} \quad p(c,x_1,\ldots,x_n) \end{aligned}$$

Thus, the classification problem is reduced to compute joint probability for every value of the class variable with the values for the predictive variables given by the instance to by classified.

When we deal with a classifier based on a Bayesian network we can use joint probability factorization shown in equation 1 in order to recover these probabilities. Initially inference with dependency networks must be done by means of Gibbs sampling, but due to the fact that we focus on classification, and under the assumption of complete data (i.e. no missing data neither in the training set nor in the test set), we can avoid Gibbs sampling. This result comes from the fact that we can use *modified ordered Gibbs sampling* as designed in (Heckerman et al., 2000). This way of doing inference over dependency networks is based on decomposing the inference task into a set of inference tasks on single variables. For example, if we consider a simple classifier based on dependency network with two predictive variables, which show dependency between them, so with the following graphical structure: $X_1 \longleftrightarrow X_2; C \longrightarrow X_1; C \longrightarrow X_2$; its probability model will be:

$$P(C, X_1, X_2) = P(C)P(X_1|C, X_2)P(X_2|C, X_1).$$

With this method joint probability can be obtained by computing each term separately. The determination of the first term does not requires Gibbs sampling. The second and third terms should be determined by using Gibbs sampling but in this case we know all values for their conditioning variables, so they can be determined directly too by looking at their conditional probability tables. If we assume that the class variable will be determined before other variables (as it is the case), by this procedure we can compute any joint probability avoiding the Gibbs sampling process.

## 4 Finding dependencies

In this section we present our algorithm to build a classifier model from data based on dependency networks. Our idea is to use independence tests in order to find dependencies between predictive variables $X_i$ and $X_j$, but taking into account the class variable, and in this way extend the Naive Bayes structure. It is known that statistic

$$2 \cdot Nc \cdot I(X_i; X_j | C = c)$$

follows a $\chi^2$ distribution with $(|X_i|-1)\cdot(|X_j|-1)$ degrees of freedom under the null hypothesis of independence, where $Nc$ is the number of input instances in which $C = c$ and $I(X_i; X_j|C = c)$ is the mutual information between variables $X_i$ and $X_j$ when the class variable $C$ is equal to $c$. Using this expression, for any variable can be found all other variables (in)dependent given the class.

But it is not this set what we are looking for, we need a set of variables which made that variable independent from the other, that is, its Markov blanket. So, we try to discover this set by carrying out an iterative process, for every predictive variable independently, in which in each step is selected as a new parent the variable not still chosen which shows more dependency with the variable under study ($X_i$). This selection is done considering all the parents previously chosen. Thus the statistic used actually is

$$2 \cdot Nc \cdot I(X_i; X_j | C = c, \mathbf{Pa}_i),$$

where $\mathbf{Pa}_i$ is the current set of $X_i$'s parents minus $C$. We assess the goodness of every candidate parent by the difference between the percentile of the $\chi^2$ distribution (with $\alpha = 0.025$) and the statistic value. Here the degrees of freedom must be

$$df = (|X_i| - 1) \cdot (|X_j| - 1) \prod_{Pa_i \in \mathbf{Pa}_i} (|Pa_i|).$$

Obviously this process finishes when all candidate parents not selected yet shows independence with $X_i$ according to this test, and in order to make search more efficient, every candidate parent that appear independent in any step in the algorithm, is rejected and is not taken into account in next iterations. The reason of rejecting these variables is just for efficiency, although we know that any of these rejected variables can become dependent in posterior interations.

Figure 1 shows the pseudo-code of this algorithm called ChiSqDN.

```
Initialize structure to Naive Bayes

For each variable X_i
  Pa_i = ∅
  Cand = X \ {X_i} // Candidate parents
  While Cand ≠ ∅
    For each variable X_j ∈ Cand
      Let val be assessment for X_j by χ² test
      If val ≤ 0 Then
        Remove X_j from Cand

    Let X_max be the best variable found
    Pa_i = Pa_i ∪ X_max
    Remove X_max from Cand

  For each variable Pa_i ∈ Pa_i
    Make new link Pa_i → X_i
```

Figure 1: Pseudo-code for ChiSqDN algorithm.

It is clear that determining degrees of freedom is critical in order to perform properly this test. The scheme outlined before is the general way for assessing this parameter, nonetheless, in some cases, especially for deterministic or almost deterministic relationships, the tests carried out using degrees of freedom in this way can fail. Deterministic relationships are pretty

common in automatic learning when input cases are relatively few. Examining the datasets used in the experiments (table 1) can be seen that there are some of them with a low number of instances. So, to make the tests in our algorithm properly, we use a method for computing degrees of freedom and handling this kind of relationships proposed in (Yilmaz et al., 2002). Basically and considering discrete variables and its conditional probability tables (CPT), this proposal reduces degrees of freedom based on the level of determinism and this level is determined by the number of columns or rows which are full of zeros.

For example, if we wanted to compute degrees of freedom for variables $X_i$ and $X_j$ whose probability distribution is shown in figure 2, we should use $(3-1)\cdot(3-1)=4$ as value for this parameter, as both variables have 3 states. Nonetheless, is clear that the relation between these two variables is not probabilistic but deterministic, because is similar to consider that variable $X_i$ has only one state. Thus, if we use this new way for computing degrees of freedom we have to take into account that this table has two rows and a column full of zeros. Then the degrees of freedom will be $(3-2-1)\cdot(3-1-1)=0$.

| | $X_j$ | | | MT |
|---|---|---|---|---|
| $X_i$ | 13 | 16 | 0 | 29 |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |
| MT | 13 | 16 | 0 | |

Figure 2: Probability table for variables $X_i$ and $X_j$.

In these cases, i. e, when the degrees of freedom are zero, it does not make sense perform the test because in this case it always holds. In (Yilmaz et al., 2002), the authors propose skipping the test and mark the relation in order to be handled by human experts. In our case, when this algorithm yields degrees of freedom equal to zero we prefer an automatic approach that is to accept dependence between tested variables if statistic value is greater than zero.

## 5 Experimental results

In order to evaluate our proposed algorithm and to measure their quality, we have selected a set of datasets from the UCI repository (Newman et al., 1998). These datasets are described in table 1. We have preprocessed these datasets in order to remove missing values and to discretize continuous variables. Missing values for each variable have been replaced by the mean or mode depending on whether it is a continuous o discrete variable respectively. For discretization we have used the algorithm proposed in (Fayyad and Irani, 1993), which is especially suggested for classification.

| Datasets | inst. | attrib. | $\|C\|$ | cont.? | missing? |
|---|---|---|---|---|---|
| australian | 690 | 15 | 2 | yes | no |
| heart | 270 | 14 | 2 | yes | no |
| hepatitis | 155 | 20 | 2 | yes | yes |
| iris | 150 | 5 | 3 | yes | no |
| lung | 32 | 57 | 3 | no | yes |
| pima | 768 | 9 | 2 | yes | no |
| post-op | 90 | 9 | 3 | yes | yes |
| segment | 2310 | 20 | 7 | yes | no |
| soybean | 683 | 36 | 19 | no | yes |
| vehicle | 846 | 19 | 4 | yes | no |
| vote | 435 | 17 | 2 | no | yes |

Table 1: Description of the datasets used in the experiments.

We have chosen a pair of algorithms from the state-of the-art in Bayesian classification which allow relationships between predictive variables. One of this algorithms is the one known as TAN (Tree Augmented Naive Bayes). The other employs a general Bayesian network in order to represent relationships between variables and is based on BAN model (Bayesian network Augmented Naive Bayes (Friedman et al., 1997)). In our experiments we use B algorithm (Buntine, 1994) as learning algorithm with BIC (Schwarz, 1978) metric to guide the search. Both algorithms use mutual information as base statistic in order to determine relationships, directly in TAN algorithm and a penalized version in BIC metric.

The experimentation process consists on running each algorithm for each database in a 5x2 cross validation. This kind of validation is an extension of the well known k-fold cross vali-

dation in which a 2-fold cv is performed five times and in any repetition a different partition is used. With this process we have the validation power from the 10-fold cv but with less correlation between each result (Dietterich, 1998).

Once we have performed our experiments in that way, we have the results for each classifier in table 2.

| | TAN | BAN+BIC | ChiSqDN |
|---|---|---|---|
| australian | 0.838841 | **0.866957** | 0.843478 |
| heart | 0.819259 | **0.826667** | 0.823704 |
| hepatitis | 0.852930 | **0.872311** | 0.865801 |
| iris | 0.948000 | **0.962667** | **0.962667** |
| lung | **0.525000** | **0.525000** | **0.525000** |
| pima | 0.772135 | **0.773177** | 0.744531 |
| post_op | 0.653333 | **0.673333** | **0.673333** |
| segment | **0.936970** | 0.909437 | 0.931169 |
| soybean | 0.898661 | **0.906893** | 0.903950 |
| vehicle | **0.719385** | 0.697163 | 0.698345 |
| vote | **0.945299** | 0.944840 | 0.931045 |

Table 2: Classification accuracy estimated for algorithms tested.

### 5.1 Analysis of the results

The analysis done for the previous results consists on carring out a statistical test. The test selected is the paired Wilcoxon signed ranks, that is a non parametric test which examines two samples in order to decide whether the differences shown by them can be assumed as not significant. In our case, if this test holds, then we can conclude that the classifiers which yield the results analysed have similar classification power.

Thus, this test is done with a level of significance of 0.05 ($\alpha = 0.05$) and it shows that both reference classifiers do not differ significantly from our classifier learned with ChiSqDN algorithm. When we compare our algorithm with TAN we obtain a p-value of 0.9188, and for the comparison with BAN model the p-value is 0.1415. Therefore ChiSqDN algorithm is as good as the Bayesian classifier considered, in spite of the approximation introduced in the factorization of the joint probability distribution due to the use of general dependency networks. Nonetheless, also as consequence of the use of dependency networks we can take advantage of the ease of learning and possibility of doing this process in parallel without introduce an extra workload.

Apart from these characteristics that can be exploited from a dependency network, we can focus on its descriptive capability, i.e. how good is the graph for each model to show relations over the domain. Taking into account pima dataset, in figures 3 and 4 are shown the graphs yielded by the algorithms BAN+BIC and ChiSqDN respectively.

Obviously, for every variable in these graphs, those that form its Markov blanket are the more informative or more related. The difference is that for a Bayesian network graph the Markov blanket set is formed by parents, children and parent of children for every variable, but for a dependency network graph the Markov blanket set are all variables directly connected. Thus in tables 3 and 4 we show the Markov blanket extracted from the corresponding graph for every variable. Evidently the sets do not match due to the automatic learning process but are quite similar. Apart from that, it is clear that discovering these relationships from the dependency network graph is easier, especially for people not used to deal with Bayesian networks, than doing it from a Bayesian network. We think that this point is an important feature of dependency networks because can lead us to a tuning process helped by human experts after the automatic learning.

| Variable | Markov blanket |
|---|---|
| mass | skin, class |
| skin | age, mass, insu, class |
| insu | skin, plas, class |
| plas | insu, class |
| age | pres, preg, skin, class |
| pres | age, class |
| preg | age, class |
| pedi | class |

Table 3: Markov blanket for each predictive variable from the BAN+BIC classifier.

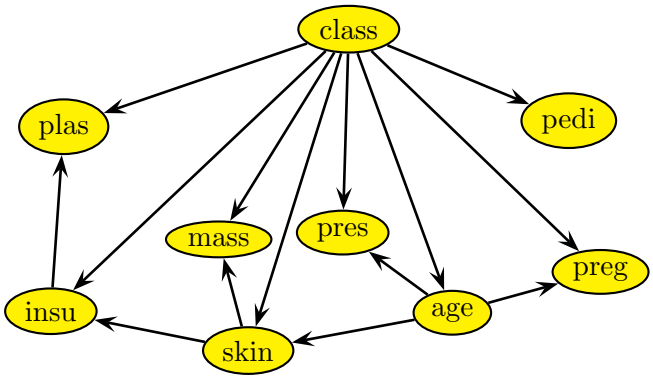



Figure 3: Network yielded by BAN algorithm with BIC metric for the pima dataset.

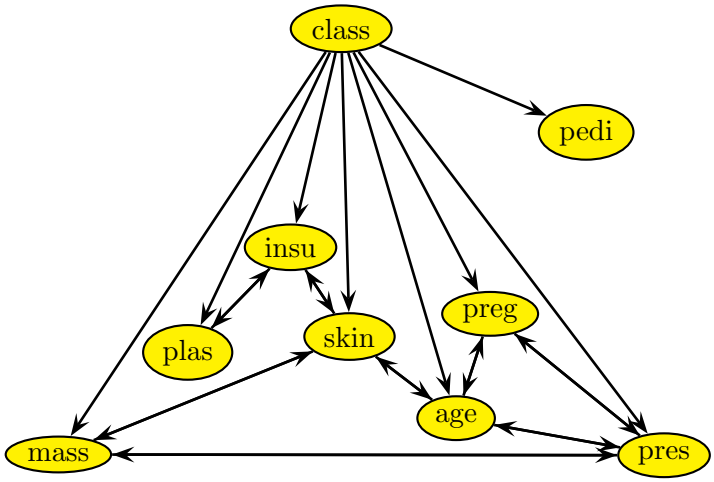



Figure 4: Network yielded by ChiSqDN algorithm for the pima dataset.

| Variable | Markov blanket |
|---|---|
| mass | skin, pres, class |
| skin | insu, mass, age, class |
| insu | plas, skin, class |
| plas | insu, class |
| age | pres, preg, skin, class |
| pres | age, mass, preg, class |
| preg | age, pres, class |
| pedi | class |

Table 4: Markov blanket for each predictive variable from the ChiSqDN classifier.

## 6 Conclusions and Future Work

We have shown how dependency networks can be used as a model for Bayesian classifiers beyond their initial purpose. We have presented a new learning algorithm which use independence tests in order to find the structural model for a classifier based on dependency networks. By means of the analysis developed in this paper, we have shown that the proposed classifier model based on dependency networks can be as good as some classifiers based on Bayesian networks. Besides this important feature, classifiers obtained by our algorithm are visually easier to understand. We mean that relationships automatically learned by the algorithm are easily understood if the classifier is modelled with a dependency network. This characteristic is especially valuable because for people is often difficult discover all relationships inside a Bayesian networks if they do not know their theory, however these relationships are clearly shown in a dependency network. Besides, we can not forget the main contributions of dependency networks: ease of learning and possibility to perform this learning in parallel.

The ChiSqDN algorithm shown here learn structural information for every variable independently, so we plan to study a parallelized version in the future. Also we will try to improve this algorithm, in terms of complexity, by using simpler statistics. Our idea is to use an approximation for the mutual information over multiples variables using a decomposition in simpler terms (Roure Alcobé, 2004).

**Acknowledgments**

This work has been partially supported by Spanish Ministerio de Ciencia y Tecnología, Junta de Comunidades de Castilla-La Mancha and European Social Fund under projects TIN2004-06204-C03-03 and PBI-05-022.

## References

Wray L. Buntine. 1994. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research*, 2:159–225.

Thomas G. Dietterich. 1998. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

R. O. Duda and P. E. Hart. 1973. *Pattern classification and scene analisys*. John Wiley and Sons.

U. Fayyad and K. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In Morgan and Kaufmann, editors, *International Joint conference on Artificial Intellegence (IJCAI)*, pages 1022–1029.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. 1997. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163.

David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. 2000. Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *Journal of Machine Learning Research*, 1:49–75.

Finn V. Jensen. 2001. *Bayesian networks and decision graphs*. Springer.

P. Langley, W. Iba, and K. Thompson. 1992. An Analisys of bayesian Classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 223–228.

Juan L. Mateo. 2006. Dependency networks: Use in automatic classification and Estimation of Distribution Algorithms (in spanish). University of Castilla-La Mancha.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. 1998. UCI Repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Josep Roure Alcobé. 2004. An approximated MDL score for learning Bayesian networks. In *Proceedings of the Second European Workshop on Probabilistic Graphical Models 2004 (PGM'04)*.

Gideon Schwarz. 1978. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464.

Yusuf Kenan Yilmaz, Ethem Alpaydin, Levent Akin, and Taner Bilgi. 2002. Handling of Deterministic Relationships in Constraint-based Causal Discovery. In *First European Workshop on Probabilistic Graphical Models (PGPM02)*, pages 204–211.

# Unsupervised naive Bayes for data clustering with mixtures of truncated exponentials

José A. Gámez
Computing Systems Department
Intelligent Systems and Data Mining Group – $i^3\mathcal{A}$
University of Castilla-La Mancha
Albacete, 02071, Spain

Rafael Rumí and Antonio Salmerón
Department of Statistics and Applied Mathematics
Data Analysis Group
University of Almería
Almería, 04120, Spain

**Abstract**

In this paper we propose a naive Bayes model for unsupervised data clustering, where the class variable is hidden. The feature variables can be discrete or continuous, as the conditional distributions are represented as mixtures of truncated exponentials (MTEs). The number of classes is determined using the *data augmentation* algorithm. The proposed model is compared with the conditional Gaussian model for some real world and synthetic databases.

## 1 Introduction

Unsupervised classification, frequently known as clustering, is a descriptive task with many applications (pattern recognition, ...) and that can be also used as a preprocessing task in the so-called knowledge discovery from data bases (KDD) process (population segmentation and outlier identification).

*Cluster analysis* or *clustering* (Anderberg, 1973; Jain et al., 1999) is understood as a decomposition or partition of a data set into groups in such a way that the objects in one group are similar to each other but as different as possible from the objects in other groups. Thus, the main goal of cluster analysis is to detect whether or not the general population is heterogeneous, that is, whether the data fall into distinct groups.

Different types of clustering algorithms can be found in the literature depending on the type of approach they follow. Probably the three main approaches are: partition-based clustering, hierarchical clustering, and probabilistic model-based clustering. From them, the first two approaches yield a *hard* clustering in the sense that clusters are exclusive, while the third one yield a *soft* clustering, that is, an object can belong to more than one cluster following a probability distribution.

In this paper we focus on probabilistic clustering, and from this point of view the data clustering problem can be defined as the inference of a probability distribution for the given database. In this work we allow nominal and numeric variables in the data set, and the novelty of the approach lies in the use of MTE (*Mixture of Truncated Exponential*) (Moral et al., 2001) densities to model the numeric variables. This model has been shown as a clear alternative to the use of Gaussian models in different tasks (inference, classification and learning) but to our knowledge its applicability to clustering remains to be studied. This is precisely the goal of this paper, to do an initial study of applying the MTE model to the data clustering problem.

The paper is structured as follows: first, Sections 2 and 3 give, respectively, some preliminaries about probabilistic clustering and mixtures of truncated exponentials. Section 4 describes the method we have developed in order to obtain a clustering from data by using mixtures of truncated exponentials to model numerical variables. Experiments over several datasets are described in Section 5. Finally, and having in mind that this is a first approach, in Section 6 we discuss about some improvements and applications to our method, and also our conclusions are presented.

## 2 Probabilistic model-based clustering

The usual way of modeling data clustering in a probabilistic approach is to add a hidden random variable to the data set, i.e., a variable whose value has been missed in all the records. This hidden variable, normally referred to as the *class* variable, will reflect the cluster membership for every case in the data set.

From the previous setting, probabilistic model-based clustering is modeled as a mixture of models (see e.g. (Duda et al., 2001)), where the states of the hidden *class* variable correspond to the components of the mixture (the number of clusters), and the multinomial distribution is used to model discrete variables while the Gaussian distribution is used to model numeric variables. In this way we move to a problem of learning from unlabeled data and usually the EM algorithm (Dempster et al., 1977) is used to carry out the learning task when the graphical structure is fixed and *structural* EM (Friedman, 1998) when the graphical structure also has to be discovered (Peña et al., 2000). In this paper we focus on the simplest model with fixed structure, the so-called *Naive Bayes* structure (fig. 1) where the class is the only root variable and all the attributes are conditionally independent given the class.

Once we decide that our graphical model is fixed, the clustering problem *reduces* to take a dataset of instances and a previously specified number of clusters ($k$), and work out each

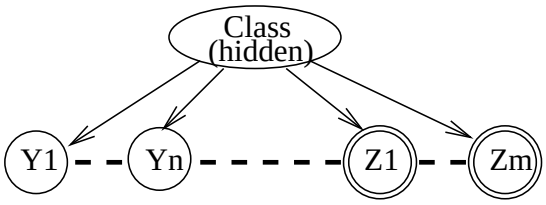



Figure 1: Graphical structure of the model. $Y_1, \ldots, Y_n$ represent discrete/nominal variables while $Z_1, \ldots, Z_m$ represent numeric variables.

cluster's distribution (multinomial or gaussian) and the population distribution between the clusters. To obtain these parameters the EM (*expectation-maximization*) algorithm is used. The EM algorithm works iteratively by carrying out the following two-steps at each iteration:

- Expectation.- Estimate the distribution of the hidden variable $C$, conditioned on the current setting of the parameter vector $\theta^k$ (i.e. the parameters needed to specify the non-hidden variables distributions).

- Maximization.- Taking into account the new distribution of $C$, use maximum-likelihood to obtain a new set of parameters $\theta^{k+1}$ from the observed data.

The algorithm starts from a given initial starting point (e.g. random parameters for $C$ or $\theta$) and under fairly general conditions it is guaranteed to converge to (at least) a local maximum of the log-likelihood function.

Iterative approaches have been described in the literature (Cheeseman and Stutz, 1996) in order to discover also the number of clusters (components of the mixture). The idea is to start with $k = 2$ and use EM to fit a model, then the algorithm tries with $k = 3, 4, ...$ while the log-likelihood improves by adding a new component (cluster) to the mixture. Of course some criteria has to be used in order to prevent overfitting.

To finish with this section we discuss about *how to evaluate the obtained clustering?*. Obviously, a clustering is useful if it produces some interesting insight in the problem we are analyzing. However, in practice and specially in order

to test new algorithms we have to use a different way of measuring the quality of a clustering. In this paper and because we produce a probabilistic description of the dataset, we use the log-likelihood (logL) of the dataset ($\mathbf{D}$) given the model to score a given clustering:

$$logL = \sum_{\mathbf{x} \in \mathbf{D}} \log p(\mathbf{x}|\{\theta, C\}).$$

## 3 Mixtures of Truncated Exponentials

The MTE model is an alternative to Gaussian models, and can be seen as a generalization to discretization models. It is defined by its corresponding potential and density as follows (Moral et al., 2001):

**Definition 1.** (MTE potential) Let $\mathbf{X}$ be a mixed $n$-dimensional random vector. Let $\mathbf{Y} = (Y_1, \ldots, Y_d)$ and $\mathbf{Z} = (Z_1, \ldots, Z_c)$ be the discrete and continuous parts of $\mathbf{X}$, respectively, with $c + d = n$. A function $f : \Omega_{\mathbf{X}} \mapsto \mathbb{R}_0^+$ is a *Mixture of Truncated Exponentials potential (MTE potential)* if one of the next conditions holds:

i. $\mathbf{Y} = \emptyset$ and $f$ can be written as

$$f(\mathbf{x}) = f(\mathbf{z}) = a_0 + \sum_{i=1}^{m} a_i \exp\left\{\sum_{j=1}^{c} b_i^{(j)} z_j\right\} \quad (1)$$

for all $\mathbf{z} \in \Omega_{\mathbf{Z}}$, where $a_i$, $i = 0, \ldots, m$ and $b_i^{(j)}$, $i = 1, \ldots, m$, $j = 1, \ldots, c$ are real numbers.

ii. $\mathbf{Y} = \emptyset$ and there is a partition $D_1, \ldots, D_k$ of $\Omega_{\mathbf{Z}}$ into hypercubes such that $f$ is defined as

$$f(\mathbf{x}) = f(\mathbf{z}) = f_i(\mathbf{z}) \quad \text{if} \quad \mathbf{z} \in D_i \ ,$$

where each $f_i$, $i = 1, \ldots, k$ can be written in the form of equation (1).

iii. $\mathbf{Y} \neq \emptyset$ and for each fixed value $\mathbf{y} \in \Omega_{\mathbf{Y}}$, $f_{\mathbf{y}}(\mathbf{z}) = f(\mathbf{y}, \mathbf{z})$ can be defined as in ii.

**Definition 2.** (MTE density) An MTE potential $f$ is an *MTE density* if

$$\sum_{\mathbf{y} \in \Omega_{\mathbf{Y}}} \int_{\Omega_{\mathbf{Z}}} f(\mathbf{y}, \mathbf{z}) d\mathbf{z} = 1 \ .$$

A univariate MTE density $f(x)$, for a continuous variable $X$, can be learnt from a sample as follows (Romero et al., 2006):

1. Select the maximum number, $s$, of intervals to split the domain.
2. Select the maximum number, $t$, of exponential terms on each interval.
3. A Gaussian kernel density is fitted to the data.
4. The domain of $X$ is split into $k \leq s$ pieces, according to changes in concavity/convexity or increase/decrease in the kernel density.
5. In each piece, an MTE potential with $m \leq t$ exponential terms is fitted to the kernel density by iterative least squares estimation.
6. Update the MTE coefficients to integrate up to one.

When learning an MTE density $f(y)$ from a sample, if $Y$ is discrete, the procedure is:

1. For each state $y_i$ of $Y$, compute the Maximium Likelihood Estimator of $p(y_i)$, which is $\dfrac{\#y_i}{\text{sample size}}$.

A *conditional MTE density* can be specified by dividing the domain of the conditioning variables and specifying an MTE density for the conditioned variable for each configuration of splits of the conditioning variables (Romero et al., 2006).

In this paper, the conditional MTE densities needed are $f(x|y)$, where $Y$ is the class variable, which is discrete, and $X$ is either a discrete or continuous variable, so the potential is composed by an MTE potential defined for $X$, for each state of the conditioning variable $Y$.

**Example 1.** The function $f$ defined as

$$f(x|y) = \begin{cases} 5 + e^{2x} + e^{x}, & y = 0, \\ & 0 < x < 2, \\ 1 + e^{-x} & y = 0, \\ & 2 \leq x < 3, \\ \frac{1}{5} + e^{-2x} & y = 1, \\ & 0 < x < 2, \\ 4e^{3x} & y = 1, \\ & 2 \leq x < 3. \end{cases}$$

is an MTE potential for a conditional $x|y$ relation, with $X$ continuous, $x \in (0, 3]$. Observe that this potential is not actually a conditional density. It must be normalised to integrate up to one.

In general, an MTE conditional density $f(x|y)$ with one conditioning discrete variable, with states $y_1, \ldots, y_s$, is learnt from a database as follows:

1. Divide the database in $s$ sets, according to variable $Y$ states.

2. For each set, learn a marginal density for $X$, as shown above, with $k$ and $m$ constant.

## 4 Probabilistic clustering by using MTEs

In the clustering algorithm we propose here, the conditional distribution for each variable given the class is modeled by an MTE density. In the MTE framework, the domain of the variables is split into pieces and in each resulting interval an MTE potential is fitted to the data. In this work we will use the so-called $five$-parameter MTE, which means that in each split there are at most five parameters to be estimated from data:

$$f(x) = a_0 + a_1 e^{a_2 x} + a_3 e^{a_4 x} \ . \qquad (2)$$

The choice of the $five$-parameter MTE is motivated by its low complexity and high fitting power (Cobb et al., 2006). The maximum number of splits of the domain of each variable has been set to four, again according to the results in the cited reference.

We start the algorithm with two clusters with the same probability, i.e., the hidden class variable has two states and the marginal probability of each state is 1/2. The conditional distribution for each feature variable given each of the two possible values of the class is initially the same, and is computed as the marginal MTE density estimated from the train database according to the estimation algorithm described in section 3.

The initial model is refined using the *data augmentation* method (Tanner and Wong, 1987):

1. First of all, we divide the train database into two parts, one of them properly for training and the other one for testing the intermediate models.

2. For each record in the test and train databases, the distribution of the class variable is computed.

3. According to the obtained distribution, a value for the class variable is simulated and inserted in the corresponding cell of the database.

4. When a value for the class variable for all the records has been simulated, the conditional distributions of the feature variables are re-estimated using the method described in section 3, using as sample the train database.

5. The log-likelihood of the new model is computed from the test database. If it is higher than the log-likelihood of the initial model, the process is repeated. Otherwise, the process is stopped, returning the best model found for two clusters.

In order to avoid long cycles, we have limited the number of iterations in the procedure above to 100. After obtaining the best model for two clusters, the algorithm continues increasing the number of clusters and repeating the procedure above for three clusters, and so on. The

algorithm continues to increase the number of clusters while the log-likelihood of the model, computed from the test database, is increased.

A model for $n$ clusters in converted into a model for $n+1$ clusters as follows:

1. Let $c_1, \ldots, c_n$ be the states of the class variable.
2. Add a new state, $c_{n+1}$ to the class variable.
3. Update the probability distribution of the class variable by re-computing the probability of $c_n$ and $c_{n+1}$ (the probability of the other values remains unchanged):
   - $p(c_n) := p(c_n)/2$.
   - $p(c_{n+1}) := p(c_n)/2$.
4. For each feature variable $X$,
   - $f(x|c_{n+1}) := f(x|c_n)$.

At this point, we can formulate the clustering algorithm as follows. The algorithm receives as argument a database with variables $X_1, \ldots, X_n$ and returns a naive Bayes network with variables $X_1, \ldots, X_n, C$, which can be used to predict the class of any object characterised by features $X_1, \ldots, X_n$.

**Algorithm CLUSTER($D$)**

1. Divide the train database $D$ into two parts: one with the 80% of the records in $D$ selected at random, for estimating the parameters ($D_p$) and another one with the remaining 20% ($D_t$) for testing the current model.
2. Let *net* be the initial model obtained from $D_p$ as described above.
3. $bestNet := net$.
4. $L_0$ :=log-likelihood($net$,$D_t$).
5. Refine *net* by *data augmentation* from database $D_p$.
6. $L_1$ :=log-likelihood($net$,$D_t$).
7. WHILE ($L_1 > L_0$)
   (a) $bestNet := net$.
   (b) $L_0 := L_1$.
   (c) Add a cluster to *net*.
   (d) Refine *net* by *data augmentation*.
   (e) $L_1$ :=log-likelihood($net$,$D_t$).
8. RETURN($bestNet$)

## 5 Experimental evaluation

In order to analyse the performance of the proposed algorithm in the data clustering task, we have selected a set of databases[1] (see Table 1) and the following experimental procedure has been carried out:

- For comparison we have used the EM algorithm implemented in the WEKA data mining suite (Witten and Frank, 2005). This is a classical implementation of the EM algorithm in which discrete and numerical variables are allowed as input attributes. Numerical attributes are modeled by means of Gaussian distributions. In this implementation the user can specify, beforehand, the number of clusters or the EM can decide how many clusters to create by cross validation. In the last case, the number of clusters is initially set to 1, the EM algorithm is performed by using a 10 folds cross validation and the log-likelihood (averaged the 10 folds) is computed. Then, the number of clusters in increased by one and the same process is carried out. If the log-likelihood with the new number of clusters has increased with respect to the previous one, the execution continues, otherwise the clustering obtained in the previous iteration is returned.
- We have divided all the data sets into train (80% of the instances) and test (20% of the instances). Then the training set has been used to learn the model but the test set is used to assess the quality of the final model, i.e., to compute the log-likelihood.

[1] In those datasets usually used for classification the class variable has been removed.

| DataSet | #instances | #attributes | discrete? | classes | source |
|---|---|---|---|---|---|
| returns | 113 | 5 | no | – | (Shenoy and Shenoy, 1999) |
| pima | 768 | 8 | no | 2 | UCI(Newman et al., 1998) |
| bupa | 345 | 6 | no | 2 | UCI(Newman et al., 1998) |
| abalone | 4177 | 9 | 2 | – | UCI(Newman et al., 1998) |
| waveform | 5000 | 21 | no | 3 | UCI(Newman et al., 1998) |
| waveform-n | 5000 | 40 | no | 3 | UCI(Newman et al., 1998) |
| net15 | 10000 | 15 | 2 | – | (Romero et al., 2006) |
| sheeps | 3087 | 23 | 5 | – | real farming data (Gámez, 2005) |

Table 1: Brief description of the data sets used in the experiments. The column discrete means if the dataset contains discrete/nominal variables (if so the number of discrete variables is given). The column classes shows the number of class-labels when the dataset comes from classification task.

- Two independent runs have been carried out for each algorithm and data set:
  - The algorithm proposed in this paper is executed over the given training set. Let $\#_{mte}$clusters be the number of clusters it has decided. Then, the EM algorithm is executed receiving as inputs the same training set and number of clusters equal to $\#_{mte}$clusters.
  - The WEKA EM algorithm is executed over the given training set. Let $\#_{em}$clusters be the number of clusters it has decided. Then, our algorithm is executed receiving as inputs the same training set and number of clusters equal to $\#_{em}$clusters.

  In this way we try to compare the performance of both algorithms over the same number of clusters.

Table 2 shows the results obtained. Out of the 8 databases used in the experiments, 7 refer to real world problems, while the other one (Net15) is a randomly generated network with joint MTE distribution (Romero et al., 2006). A rough analysis of the results suggest a better performance of the EM clustering algorithm: in 5 out of the 8 problems, it obtains a model with higher log-likelihood than the MTE-based algorithm. However, it must be pointed out that the number of clusters contained in the best models generated by the MTE-based algorithm is significantly lower than the number of clusters produced by the EM. In some cases, a model with many clusters, all of them with low probability, is not as useful as a model with fewer clusters but with higher probability, even if the last one has lower likelihood.

Another fact that must be pointed out is that, in the four problems where the exact number of classes is known (pima, bupa, waveform and waveform-n) the number of clusters returned by the MTE-based algorithms is more accurate. Besides, forcing the number of clusters in the EM algorithm to be equal to the number of clusters generated by the MTE method, the differences turn in favor of the MTE-based algorithm.

After this analysis, we consider the MTE-based clustering algorithm a promising method. It must be taken into account that the EM algorithm implemented in Weka is very optimised and sophisticated, while the MTE-based method presented here is a first version in which many aspects are still to be improved. Even in these conditions, our method is competitive in some problems. Models which are away from normality (as Net15) are more appropriate for applying the MTE-based algorithm rather than EM.

## 6 Discussion and concluding remarks

In this paper we have introduced a novel unsupervised clustering algorithm which is based on the MTE model. The model has been tested

| DataSet | $\#_{mte}$clusters | MTE | EM | $\#_{em}$clusters | MTE | EM |
|---|---|---|---|---|---|---|
| returns | 2 | 212 | 129 | – | – | – |
| Pima | 3 | -4298 | -4431 | 7 | -4299 | -3628 |
| bupa | 5 | -1488 | -1501 | 3 | -1491 | -1489 |
| abalone | 11 | 6662 | 7832 | 15 | 6817 | 8090 |
| waveform | 3 | -38118 | -33084 | 11 | -38153 | -31892 |
| Waveform-n | 3 | -65200 | -59990 | 11 | -65227 | -58829 |
| Net15 | 4 | -3418 | -6014 | 23 | -3723 | -5087 |
| Sheeps | 3 | -34214 | -35102 | 7 | -34215 | -34145 |

Table 2: Results obtained. Columns 2-4 represents the number of clusters discovered by the proposed MTE-based method, its result and the result obtained by EM when using that number of clusters. Columns 5-7 represents the number of clusters discovered by EM algorithm, its result and the result obtained by the MTE-based algorithm when using that number of clusters.

over 8 databases, 7 of them corresponding to real-world problems and some of them commonly used for benchmarks. We consider that the results obtained, in compares with the EM algorithm, are at least promising. The algorithm we have used in this paper can still be improved. For instance, during the model construction we have used a train-test approach for estimating the parameters and validating the intermediate models obtained. However, the EM algorithm uses cross validation. We plan to include 10-fold cross validation in a forthcoming version of the algorithm.

Another aspect that can be improved is the way in which the model is updated when the number of clusters is increased. We have followed a very simplistic approach: for the feature variables, we duplicate the same density function conditional on the previous last cluster, while for the class variable, we divide the probability of the former last cluster with the new one. We think that a more sophisticated approach, based on component splitting and parameter reusing similar to the techniques proposed in (Karciauskas, 2005) could improve the performance of our algorithm.

The algorithm proposed in this paper can be applied not only to clustering problems, but also to construct an algorithm for approximate probability propagation in hybrid Bayesian networks. This idea was firstly proposed in (Lowd and Domingos, 2005), and is based on the simplicity of probability propagation in naive Bayes structures. A naive Bayes model with discrete class variable, actually represents the conditional distribution of each variable to the rest of variables as a mixture of marginal distributions, being the number of components in the mixture equal to the number of states of the class variable. So, instead of constructing a general Bayesian network from a database, if the goal is probability propagation, this approach can be competitive. The idea is specially interesting for the MTE model, since probability propagation has a high complexity for this model.

**Acknowledgments**

Acknowledgements This work has been supported by Spanish MEC under project TIN2004-06204-C03-{01,03}.

## References

M.R. Anderberg. 1973. *Cluster Analysis for Applications*. Academic Press.

P. Cheeseman and J. Stutz. 1996. Bayesian classification (AUTOCLASS): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press/MIT Press.

B.R. Cobb, P.P. Shenoy, and R. Rumí. 2006. Approximating probability density functions with mixtures of truncated exponentials. *Statistics and Computing*, In press.

A. P. Dempster, N. M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1(39):1–38.

R.O. Duda, P.E. Hart, and D.J. Stork. 2001. *Pattern Recognition*. Wiley.

N. Friedman. 1998. The Bayesian structural EM algorithm. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 129–138. Morgan Kaufmann.

J.A. Gámez. 2005. Mining the ESROM: A study of breeding value prediction in manchego sheep by means of classification techniques plus attribute selection and construction. Technical Report DIAB-05-01-3, Computer Systems Department. University of Castilla-La Mancha.

A.K. Jain, M.M. Murty, and P.J. Flynn. 1999. Data clustering: a review. *ACM COmputing Surveys*, 31(3):264–323.

G. Karciauskas. 2005. *Learning with hidden variables: A parameter reusing approach for tree-structured Bayesian networks*. Ph.D. thesis, Department of Computer Science. Aalborg University.

D. Lowd and P. Domingos. 2005. Naive bayes models for probability estimation. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 529–536. ACM Press.

S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In *Lecture Notes in Artificial Intelligence*, volume 2143, pages 135–143.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. 1998. UCI Repository of machine learning databases. www.ics.uci.edu/∼mlearn/MLRepository.html.

J.M. Peña, J.A. Lozano, and P. Larrañaga. 2000. An improved bayesian structural EM algorithm for learning bayesian networks for clustering. *Pattern Recognition Letters*, 21:779–786.

V. Romero, R. Rumí, and A. Salmerón. 2006. Learning hybrid Bayesian networks using mixtures of truncated exponentials. *Inernational Journal of Approximate Reasoning*, 42:54–68.

C. Shenoy and P.P. Shenoy. 1999. Bayesian network models of portfolio risk and return. In W. Lo Y.S. Abu-Mostafa, B. LeBaron and A.S. Weigend, editors, *Computational Finance*, pages 85–104. MIT Press, Cambridge, MA.

M.A. Tanner and W.H Wong. 1987. The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, 82:528–550.

I.H. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

# Selecting Strategies for Infinite-Horizon Dynamic LIMIDs

Marcel A.J. van Gerven
Institute for Computing and Information Sciences
Radboud University Nijmegen
Toernooiveld 1
6525 ED Nijmegen, The Netherlands

Francisco J. Díez
Department of Artificial Intelligence, UNED
Juan del Rosal 16
28040 Madrid, Spain

**Abstract**

In previous work we have introduced dynamic limited-memory influence diagrams (DLIMIDs) as an extension of LIMIDs aimed at representing infinite-horizon decision processes. If a DLIMID respects the first-order Markov assumption then it can be represented by 2TLIMIDS. Given that the treatment selection algorithm for LIMIDs, called single policy updating (SPU), can be infeasible even for small finite-horizon models, we propose two alternative algorithms for treatment selection with 2TLIMIDS. First, single rule updating (SRU) is a hill-climbing method inspired upon SPU which needs not iterate exhaustively over all possible policies at each decision node. Second, a simulated annealing algorithm can be used to avoid the local-maximum policies found by SPU and SRU.

## 1 Introduction

An important goal in artificial intelligence is to create systems that make optimal decisions in situations characterized by uncertainty. One can think for instance of a robot that navigates based on its sensor readings in order to achieve goal states, or of a medical decision-support system that chooses treatments based on patient status in order to maximize life-expectancy.

*Limited-memory influence diagrams* (LIMIDs) are a formalism for decision-making under uncertainty (Lauritzen and Nilsson, 2001). They generalize standard influence diagrams (Howard and Matheson, 1984) by relaxing the assumption that the whole observed history is taken into account when making a decision, and by dropping the requirement that a complete order is defined over decisions. This increases the size and variety of decision problems that can be handled, although possibly at the expense of finding only approximations to the optimal strategy. Often however, there is no predefined time at which the process stops (i.e. we have an *infinite-horizon decision process*) and in that case the LIMID would also become infinite in size. In previous work, we have introduced dynamic LIMIDs (DLIMIDs) and two-stage temporal LIMIDs (2TLIMIDs) as an extension of standard LIMIDs that allow for the representation of infinite-horizon decision processes (van Gerven et al., 2006). However, the problem of finding acceptable strategies for DLIMIDs has not yet been addressed. In this paper we discuss a number of techniques to approximate the optimal strategy for infinite-horizon dynamic LIMIDs. We demonstrate the performance of these algorithms on a non-trivial decision problem.

## 2 Preliminaries

### 2.1 Bayesian Networks

Bayesian networks (Pearl, 1988) provide for a compact factorization of a joint probability dis-

tribution over a set of random variables by exploiting the notion of *conditional independence*. One way to represent conditional independence is by means of an acyclic directed graph (ADG) $G$ where vertices $V(G)$ correspond to random variables $\mathbf{X}$ and the absence of arcs from the set of arcs $A(G)$ represents conditional independence. Due to this one-to-one correspondence we will use vertices $v \in V(G)$ and random variables $X \in \mathbf{X}$ interchangeably. A *Bayesian network* (BN) is defined as $\mathcal{B} = (\mathbf{X}, G, P)$, such that the joint probability distribution $P$ over $\mathbf{X}$ factorizes according to $G$:

$$P(\mathbf{X}) = \prod_{X \in \mathbf{X}} P(X \mid \pi_G(X))$$

where $\pi_G(X) = \{X' \mid (X', X) \in A(G)\}$ denotes the *parents* of $X$. We drop the subscript $G$ when clear from context. We assume that (random) variables $X$ can take values $x$ from a set $\Omega_X$ and use $\mathbf{x}$ to denote an element in $\Omega_{\mathbf{X}} = \times_{X \in \mathbf{X}} \Omega_X$ for a set $\mathbf{X}$ of (random) variables.

### 2.2 LIMIDs

Although Bayesian networks are a natural framework for probabilistic knowledge representation and reasoning under uncertainty, they are not suited for decision-making under uncertainty. Influence-diagrams (Howard and Matheson, 1984) are graphical models that extend Bayesian networks to incorporate decision-making but are restricted to the representation of small decision problems. *Limited-memory influence diagrams* (LIMIDs) (Lauritzen and Nilsson, 2001) generalize standard influence-diagrams by relaxing the *no-forgetting* assumption, which states that, given a total ordering of decisions, information present when making decision $D$ is also taken into account when making decision $D'$, if $D$ precedes $D'$ in the ordering. A LIMID is defined as a tuple $\mathcal{L} = (\mathbf{C}, \mathbf{D}, \mathbf{U}, G, P)$ consisting of the following components. $\mathbf{C}$ represents a set of random variables, called *chance variables*, $\mathbf{D}$ represents a set of *decisions* available to the decision maker, where a decision $D \in \mathbf{D}$ is defined as a variable that can take on a value from a set of choices $\Omega_D$, and $\mathbf{U}$ is a set of *utility functions*, which represent the utility of being in a certain state as defined by configurations of chance and decision variables. $G$ is an acyclic directed graph (ADG) with vertices $V(G)$ corresponding to $\mathbf{C} \cup \mathbf{D} \cup \mathbf{U}$, where we use $\mathbf{V}$ to denote $\mathbf{C} \cup \mathbf{D}$. Again, due to this correspondence, we will use nodes in $V(G)$ and corresponding elements in $\mathbf{C} \cup \mathbf{D} \cup \mathbf{U}$ interchangeably. The meaning of an arc $(X, Y) \in A(G)$ is determined by the type of $Y$. If $Y \in \mathbf{C}$ then the conditional probability distribution associated with $Y$ is conditioned by $X$. If $Y \in \mathbf{D}$ then $X$ represents information that is present to the decision maker prior deciding upon $Y$. We call $\pi(Y)$ the *informational predecessors* of $Y$. The order in which decisions are made in a LIMID should be compatible with the partial order induced by the ADG and are based only on the parents $\pi(D)$ of a decision $D$. If $Y \in \mathbf{U}$ then $X$ takes part in the specification of the utility function $Y$ such that $Y \colon \Omega_{\pi(Y)} \to \mathbb{R}$. In this paper, it is assumed that utility nodes cannot have children and the joint utility function $\mathcal{U}$ is additively decomposable such that $\mathcal{U} = \sum_{U \in \mathbf{U}} U$. $P$ specifies for each $\mathbf{d} \in \Omega_{\mathbf{D}}$ a distribution

$$P(\mathbf{C} : \mathbf{d}) = \prod_{C \in \mathbf{C}} P(C \mid \pi(C))$$

that represents the distribution over $\mathbf{C}$ when we externally set $\mathbf{D} = \mathbf{d}$ (Cowell et al., 1999). Hence, $\mathbf{C}$ is not conditioned on $\mathbf{D}$, but rather parameterized by $\mathbf{D}$, and if $\mathbf{D}$ is unbound then we write $P(\mathbf{C} : \mathbf{D})$.

A *stochastic policy* for decisions $D \in \mathbf{D}$ is defined as a distribution $P_D(D \mid \pi(D))$ that maps configurations of $\pi(D)$ to a distribution over alternatives for $D$. If $P_D$ is degenerate then we say that the policy is deterministic. A *strategy* is a set of policies $\Delta = \{P_D \colon D \in \mathbf{D}\}$ which induces the following joint distribution over the variables in $\mathbf{V}$:

$$P_\Delta(\mathbf{V}) = P(\mathbf{C} : \mathbf{D}) \prod_{D \in \mathbf{D}} P_D(D \mid \pi(D)).$$

Using this distribution we can compute the expected utility of a strategy $\Delta$ as $\mathrm{E}_\Delta(\mathcal{U}) = \sum_{\mathbf{v}} P_\Delta(\mathbf{v}) \mathcal{U}(\mathbf{v})$. The aim of any rational decision maker is then to maximize the expected utility by finding the optimal strategy $\Delta^* \equiv \arg\max_\Delta \mathrm{E}_\Delta(\mathcal{U})$.

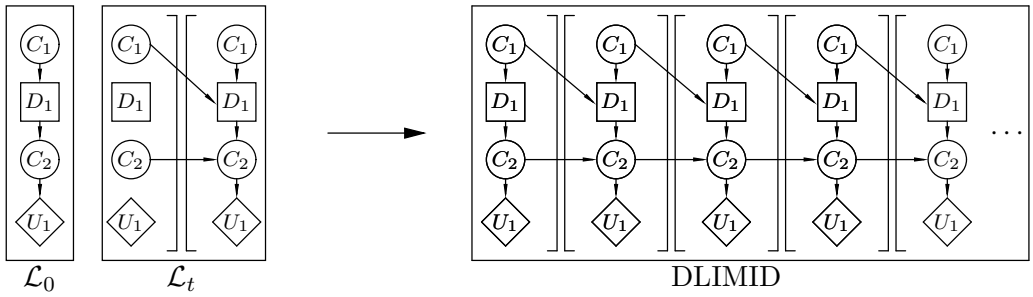



Figure 1: Chance nodes are shown as circles, decision nodes as squares and utility nodes as diamonds. The 2TLIMID (left) can be unrolled into a DLIMID (right), where large brackets denote the boundary between subsequent times. Note that due to the definition of a 2TLIMID, the informational predecessors of a decision can only occur in the same, or the preceding time-slice.

### 2.3 Dynamic LIMIDs and 2TLIMIDs

Although LIMIDs can often represent finite-horizon decision processes more compactly than standard influence diagrams, they cannot represent infinite-horizon decision processes. Recently, we introduced *dynamic* LIMIDs (DLIMIDs) as an explicit representation of decision processes (van Gerven et al., 2006). To represent time, we use $\mathbf{T} \subseteq \mathbb{N}$ to denote a set of time points, which we normally assume to be an interval $\{u \mid t \leq u \leq t', \{t, u, t'\} \subset \mathbb{N}\}$, also written as $t : t'$. We assume that chance variables, decision variables and utility functions are indexed by a superscript $t \in \mathbf{T}$, and use $\mathbf{C}^{\mathbf{T}}$, $\mathbf{D}^{\mathbf{T}}$ and $\mathbf{U}^{\mathbf{T}}$ to denote all chance variables, decision variables and utility functions at times $t \in \mathbf{T}$, where we abbreviate $\mathbf{C}^{\mathbf{T}} \cup \mathbf{D}^{\mathbf{T}}$ with $\mathbf{V}^{\mathbf{T}}$.

A DLIMID is simply defined as a LIMID $(\mathbf{C}^{\mathbf{T}}, \mathbf{D}^{\mathbf{T}}, \mathbf{U}^{\mathbf{T}}, G, P)$ such that for all pairs of variables $X^t, Y^u \in \mathbf{V}^{\mathbf{T}} \cup \mathbf{U}^{\mathbf{T}}$ it holds that if $t < u$ then $Y^u$ cannot precede $X^t$ in the ordering induced by $G$. If $\mathbf{T} = 0 : N$, where $N \in \mathbb{N}$ is the (possibly infinite) *horizon*, then we suppress $\mathbf{T}$ altogether, and we suppress indices for individual chance variables, decision variables and utility functions when clear from context. If a DLIMID respects the Markov assumption that the future is independent of the past, given the present, then it can be compactly represented by a 2TLIMID (see Fig. 1), which is a pair $\mathcal{T} = (\mathcal{L}_0, \mathcal{L}_t)$ with *prior model* $\mathcal{L}_0 = (\mathbf{C}^0, \mathbf{D}^0, \mathbf{U}^0, G^0, P^0)$ and *transition model* $\mathcal{L}_t = (\mathbf{C}^{t-1:t}, \mathbf{D}^{t-1:t}, \mathbf{U}^t, G, P)$ such that for all $V^{t-1} \in \mathbf{V}^{t-1:t}$ in the transition model it holds that $\pi_{G^t}(V^{t-1}) = \varnothing$. The transition model is not yet bound to any specific $t$, but if bound to some $t \in 1 : N$, then it is used to represent $P(\mathbf{C}^t : \mathbf{D}^{t-1:t})$ and utility functions $U \in \mathbf{U}^t$, where both $G$ and $P$ do not depend on $t$. The prior model is used to represent the initial distribution $P^0(\mathbf{C}^0 : \mathbf{D}^0)$ and utility functions $U \in \mathbf{U}^0$. The *interface* of the transition model is the set $\mathbf{I}^t \subseteq \mathbf{V}^{t-1}$ such that $(V_i^{t-1}, V_j^t) \in A(G) \Leftrightarrow V_i^{t-1} \in \mathbf{I}^t$. Given a horizon $N$, we may *unroll* a 2TLIMID for $n$ *time-slices* in order to obtain a DLIMID with the following joint distribution:

$$P(\mathbf{C}, \mathbf{D}) = P^0(\mathbf{C}^0 : \mathbf{D}^0) \prod_{t=1}^{N} P(\mathbf{C}^t : \mathbf{D}^{t-1:t}).$$

Let $\Delta^t = \{P_D^t(D \mid \pi_G(D)) \mid D \in \mathbf{D}^t\}$ denote the strategy for a time-slice $t$ and let the whole strategy be $\Delta = \Delta^0 \cup \cdots \cup \Delta^N$. Given $\Delta^0$, $\mathcal{L}_0$ defines a distribution over the variables in $\mathbf{V}^0$:

$$P_{\Delta^0}(\mathbf{V}^0) = P^0(\mathbf{C}^0 : \mathbf{D}^0) \prod_{D \in \mathbf{D}^0} P_D(D \mid \pi_{G^0}(D))$$

and given a strategy $\Delta^t$ with $t > 0$, $\mathcal{L}_t$ defines the following distribution over variables in $\mathbf{V}^t$:

$$P_{\Delta^t}(\mathbf{V}^t \mid \mathbf{I}^t) = P(\mathbf{C}^t : \mathbf{D}^u) \prod_{D \in \mathbf{D}^t} P_D(D \mid \pi_G(D))$$

with $u = t - 1 : t$. Combining both equations, given a horizon $N$ and strategy $\Delta$, a 2TLIMID induces a distribution over variables in $\mathbf{V}$:

$$P_\Delta(\mathbf{V}) = P_{\Delta^0}(\mathbf{V}^0) \prod_{t=1}^{N} P_{\Delta^t}(\mathbf{V}^t \mid \mathbf{I}^t). \quad (1)$$

Let $\mathcal{U}^0(\mathbf{V}^0) = \sum_{U \in \mathbf{U}^0} U(\pi_{G^0}(U))$ denote the joint utility for time-slice 0 and let $\mathcal{U}^t(\mathbf{V}^{t\text{-}1:t}) = \sum_{U \in \mathbf{U}^t} U(\pi_G(U))$ denote the joint utility for time-slice $t > 0$. We redefine the joint utility function for a dynamic LIMID as

$$\mathcal{U}(\mathbf{V}) = \mathcal{U}^0(\mathbf{V}^0) + \sum_{t=1}^{N} \gamma^t \mathcal{U}^t(\mathbf{V}^{t\text{-}1:t})$$

where $\gamma$ with $0 \leq \gamma < 1$ is a *discount factor*, representing the notion that early rewards are worth more than rewards earned at a later time.

In this way, we can use DLIMIDs to represent infinite-horizon Markov decision processes.

### 2.4 Memory variables

Figure 1 makes clear that the informational predecessors of a decision variable $D^t$ can only occur in time-slices $t$ or $t-1$ (viz. Eq. 1). Observations made earlier in time will not be taken into account and as a result, states that are qualitatively different can appear the same to the decision maker, which leads to suboptimal policies. This phenomenon is known as *perceptual aliasing* (Whitehead and Ballard, 1991). We try to avoid this problem by introducing *memory variables* that represent a summary of the observed history. With each observable variable $V \in \mathbf{V}$, we associate a memory variable $M \in \mathbf{C}$, such that the parents of a memory variable are given by $\pi(M^0) = \{V^0\}$ and $\pi(M^t) = \{V^t, M^{t-1}\}$ for $t \in \{1, \ldots, N\}$ and all children of $M^t$, with $t \in \{0, \ldots, N\}$, are decision variables $D \in \mathbf{D}^t$.

Figure 2 visualizes the concept of a memory variable, and is used as the running example for this paper. It depicts a 2TLIMID for the treatment of patients that may or may not have a *disease* $D$. The disease can be identified by a *finding* $F$, which is the result of a *laboratory test* $L$, having an associated cost that is captured by the utility function $U_2$. The *memory* concerning findings is represented by the memory variable $M$, and based on this memory, we decide whether or not to perform *treatment* $T$, which has an associated cost, captured by the utility function $U_3$. Memory concerning *past* findings will be used to decide whether or not to perform the laboratory test. If the patient has the disease then this decreases the chances of patient

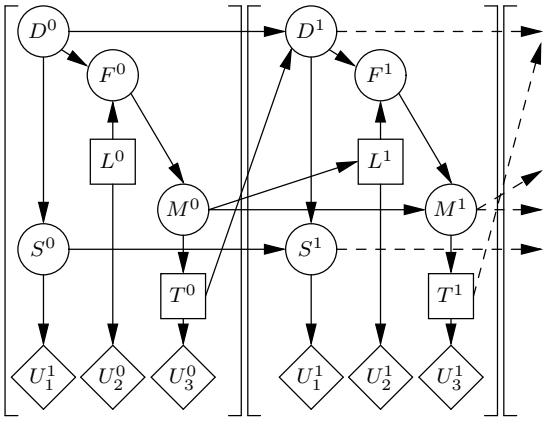



Figure 2: A DLIMID for patient treatment as specified by a 2TLIMID.

*survival* $S$. Patient survival has an associated utility $U_1$. An initial strategy, as for instance suggested by a physician, might be to *always treat* and *never test*.

There are various ways to define $\Omega_M$ and the distributions $P(M^0 \mid V^0)$ and $P(M^t \mid V^t, M^{t-1})$ for a memory variable $M$. The optimal way to define our memory variables is problem dependent, and we assume that this definition is based on the available domain knowledge. For our running example, we choose $\Omega_M = \{a, n, p\} \times \{a, n, p\} \times \{a, n, p\}$, where $a$ stands for the *absence* of a finding, $n$ for a *negative* finding, and $p$ for a *positive* finding, which is evidence for the presence of the disease. $M^t = (z, y, x)$ then denotes the current finding $x$, the finding in the previous time-slice $y$, and the finding two time-slices ago $z$. If the new finding is $F^{t+1} = f$ then $M^{t+1} = (y, x, f)$, and since we have not yet observed any findings at $t = 0$, the initial memory is $(a, a, a)$ if we did not test and $(a, a, n)$ or $(a, a, p)$ if the test was performed.

## 3 Improving Strategies in Infinite-Horizon DLIMIDs

Although DLIMIDs constructed from 2TLIMIDs can represent infinite-horizon Markov decision processes, to date, the problem of strategy improvement using 2TLIMIDs has not been addressed. In this section, we explore techniques for finding strategies with high expected utility.

### 3.1 Single Policy Updating

One way to improve strategies in standard LIMIDs is to use an iterative procedure called *single policy updating* or SPU for short (Lauritzen and Nilsson, 2001). Let $\Delta_0 = \{p_1, \ldots, p_n\}$ be an ordered set representing the initial strategy where $p_j$, $1 \leq j \leq n$ stands for a (randomly initialized) policy $P_{D_j}$ for a decision $D_j$. We say $p_j$ is the *local maximum policy* for a strategy $\Delta$ at decision $D_j$ if $\mathrm{E}_{\Delta}(\mathcal{U})$ cannot be improved by changing $p_j$. Single policy updating proceeds by iterating over all decision variables (called a cycle) to find local maximum policies, and to reiterate until no further improvement in expected utility can be achieved. SPU converges in a finite number of cycles to a *local maximum strategy* $\Delta$ where each $p_j \in \Delta$ is a local maximum policy. Note that this local maximum strategy is not necessarily the global maximum strategy $\Delta^*$.

To find local maximum policies in standard LIMIDs, Lauritzen and Nilsson (2001) use a message passing algorithm, optimized for single policy updating. In this paper, we resort to standard inference algorithms for finding strategies for (infinite-horizon) DLIMIDs. We make use of the fact that given $\Delta$, a LIMID $\mathcal{L} = (\mathbf{C}, \mathbf{D}, \mathbf{U}, G, P)$ may be converted into a Bayesian network $\mathcal{B} = (\mathbf{X}, G', P')$. Since $\Delta$ induces a distribution over variables in $\mathbf{V}$ (viz. Eq. 1), we can use $\Delta$ to convert decision variables $D \in \mathbf{D}$ to random variables $X \in \mathbf{X}$ with parents $\pi_G(D)$ such that $P(X \mid \pi_{G'}(X)) = P_D(D \mid \pi_G(D))$. Additionally, utility functions $U \in \mathbf{U}$ may be converted into random variables $X$ by means of Cooper's transformation (Cooper, 1988), which allows us to compute $\mathrm{E}_{\Delta}(\mathcal{U})$. We use $B(\mathcal{L}, \Delta)$ to denote this conversion of a LIMID into a Bayesian network.

Single policy updating cannot be applied directly to an infinite-horizon DLIMID since computing $\mathrm{E}_{\Delta}(\mathcal{U})$ would need an infinite number of steps. In order to approximate the expected utility given $\Delta$, we assume that the DLIMID can be represented as a 2TLIMID $\mathcal{T} = (\mathcal{L}_0, \mathcal{L}_t)$ and $\Delta$ can be expressed as a pair $(\Delta^0, \Delta^t)$, such that $\Delta^0$ is the strategy at $t = 0$ and $\Delta^t$ is a *stationary* strategy at $t \in 1 : \infty$. Note that the optimal strategy is deterministic and stationary for infinite-horizon and fully observable Markov decision processes (Ross, 1983). In the partially observable case, we can only expect to find approximations to the optimal strategy by using memory variables that represent part of the observational history (Meuleau et al., 1999).

SPU($\mathcal{T}$,$\Delta_0$,$\epsilon$):
  $\Delta = \Delta_0$, $euMax = \mathrm{E}^{\epsilon}_{\Delta_0}(\mathcal{U})$.
  **repeat**
    $euMaxOld = euMax$
    **for** $j = 1$ to $n$ **do**
      **for all** policies $p'_j$ for $\Delta$ at $D_j$ **do**
        $\Delta' = p'_j * \Delta$
        **if** $\mathrm{E}^{\epsilon}_{\Delta'}(\mathcal{U}) > euMax + \epsilon$ **then**
          $\Delta = \Delta'$ **and** $euMax = \mathrm{E}^{\epsilon}_{\Delta'}(\mathcal{U})$
        **end if**
      **end for**
    **end for**
  **until** $euMax = euMaxOld$
  **return** $\Delta$

Figure 3: Single policy updating for 2TLIMIDs.

We proceed by converting $(\mathcal{L}_0, \mathcal{L}_t)$ into $(\mathcal{B}_0, \mathcal{B}_t)$ with $\mathcal{B}_0 = B(\mathcal{L}_0, \Delta^0)$ and $\mathcal{B}_t = B(\mathcal{L}_t, \Delta^t)$, where $(\mathcal{B}_0, \mathcal{B}_t)$ is known as a *two-stage temporal Bayes net* (Dean and Kanazawa, 1989). We use inference algorithms that operate on $(\mathcal{B}_0, \mathcal{B}_t)$ in order to compute an approximation of the expected utility. In our work, we have used the *interface algorithm* (Murphy, 2002), for which it holds that the space and time taken to compute each $P(\mathbf{X}^t \mid \mathbf{X}^{t-1})$ does not depend on the number of time-slices. The approximation $\mathrm{E}^{\epsilon}_{\Delta}(\mathcal{U})$ is made using a finite number of time-slices $k$, where $k$ is such that $\gamma^k < \epsilon$ with $\epsilon > 0$. The discount factor $\gamma$ ensures that $\lim_{t\to\infty} \mathrm{E}_{\Delta}(\mathcal{U}) = 0$. Let $\Delta_0 = \Delta^0 \cup \Delta^t$ be the initial strategy with $\Delta^0 = \{p_1, \ldots, p_m\}$ and $\Delta^t = \{p_{m+1}, \ldots, p_n\}$, where $m$ is the number of decision variables in $\mathcal{L}_0$ and $n-m$ is the number of decision variables in $\mathcal{L}_t$. Following (Lauritzen and Nilsson, 2001), we define $p'_j * \Delta$ as the strategy obtained by replacing $p_j$ with $p'_j$ in $\Delta$. SPU based on a 2TLIMID $\mathcal{T}$ with initial strategy $\Delta_0$ is then defined by the algorithm in Fig. 3.

### 3.2 Single Rule Updating

An obstacle for the use of SPU for strategy improvement is the fact that if the state-space $\Omega_{\pi(D)}$ for informational predecessors $\pi(D)$ of a decision variable $D$ becomes large, then it becomes impossible in practice to iterate over all possible policies for $D$. The number of policies that needs to be evaluated at each decision variable $D$ grows as $k^{m^r}$, with $k = |\Omega_D|$, assuming that $|\Omega_{V_j}| = m$ for all $V_j \in \pi(D)$, and $r$ is the number of informational predecessors of $D$. For example, looking back for two time-slices within our model leads to 2 policies for $L^0$ and $2^{27}$ policies for $T^0$, $L^1$ and $T^1$, which is computationally infeasible, even for this small example.

For this reason, we introduce a hill-climbing search called *single rule updating* (SRU) that is inspired upon single policy updating. A deterministic policy can be viewed as a mapping $p_j \colon \Omega_{\pi(D_j^t)} \rightarrow \Omega_{D_j^t}$, describing for each configuration $\mathbf{x} \in \Omega_{\pi(D_j^t)}$ an action $a \in \Omega_{D_j^t}$. We call $(\mathbf{x}, a) \in p_j$ a *decision rule*. Instead of exhaustively searching over all possible policies for each decision variable, we try to increase the expected utility by local changes to the decision rules within the policy. I.e., at each step we change one decision-rule within the policy, accepting the change when the expected utility increases. We use $(\mathbf{x}, a') * p_j$ to denote the replacement of $(\mathbf{x}, a)$ by $(\mathbf{x}, a')$ in $p_j$. Similar to SPU, we keep iterating until there is no further increase in the expected utility (Fig. 4).

SRU decreases the number of policies that need to be evaluated in each *local* cycle for a decision variable to $km^r$, where notation is as before. For our example, we only need to evaluate 2 policies for $L^0$ and 54 policies for $T^0$, $L^1$ and $T^1$ in each local cycle, albeit at the expense of replacing the exhaustive search by a hill-climbing strategy, increasing the risk of ending up in local maxima, and having to run local cycles until convergence.

### 3.3 Simulated Annealing

SPU and SRU both find local maximum strategies, which may not be the optimal strategy $\Delta^*$. To see this, consider the proposed strategy for our running example (Fig. 2) to never test and always treat. Suppose this is our initial strategy $\Delta_0$ for either the SPU or SRU algorithm. Trying to improve the policy for the laboratory test $L$ we find that performing the test will only decrease the expected utility since the test has no informational value (we always treat) but does have an associated cost. Conversely, trying to improve the policy for treatment we find that the test is never performed and therefore it is more safe to always treat. Hence, SPU and SRU will stop after one cycle, returning the proposed strategy as the local optimal strategy.

SRU($\mathcal{T}$,$\Delta_0$,$\epsilon$):
  $\Delta = \Delta_0$, $euMax = \mathrm{E}^{\epsilon}_{\Delta_0}(\mathcal{U})$
  **repeat**
    $euMaxOld = euMax$
    **for** $j = 1$ to $n$ **do**
      **repeat**
        $euMaxLocal = euMax$
        **for all** configurations $\mathbf{x} \in \Omega_{\pi(D_j)}$ **do**
          **for all** actions $a' \in \Omega_{D_j}$ **do**
            $p'_j = (\mathbf{x}, a') * p_j$
            $\Delta' = p'_j * \Delta$
            **if** $\mathrm{E}^{\epsilon}_{\Delta'}(\mathcal{U}) > euMax + \epsilon$ **then**
              $\Delta = \Delta'$ **and** $euMax = \mathrm{E}^{\epsilon}_{\Delta'}(\mathcal{U})$
            **end if**
          **end for**
        **end for**
      **until** $euMax = euMaxLocal$
    **end for**
  **until** $euMax = euMaxOld$
  **return** $\Delta$

Figure 4: Single rule updating for 2TLIMIDs.

In order to improve upon the strategies found by SRU, we resort to *simulated annealing* (SA), which is a heuristic search method that tries to avoid getting trapped into local maximum solutions that are found by hill-climbing techniques such as SRU (Kirkpatrick et al., 1983). SA chooses candidate solutions by looking at neighbors of the current solution as defined by a *neighborhood function*. Local maxima are avoided by sometimes accepting worse solutions according to an *acceptance function*.

SA($\mathcal{T}$,$\Delta_0$,$\epsilon$,$\tau_0$, $T$):
  $\Delta = \Delta_0$, $t = 0$, $eu = \mathrm{E}^{\epsilon}_{\Delta}(\mathcal{U})$
  **repeat**
    select a random decision variable $D_j$
    select a random decision rule $(\mathbf{x}, a) \in p_j$
    select a random action $a' \in \Omega_{D_j}$, $a' \neq a$
    $p'_j = (\mathbf{x}, a') * p_j$
    $\Delta' = p'_j * \Delta$
    $eu' = \mathrm{E}^{\epsilon}_{\Delta'}(\mathcal{U})$
    **if** $\theta \leq P(a(\Delta') = \text{yes} \mid eu + \epsilon, eu', t)$ **then**
      $\Delta = \Delta'$ **and** $eu = eu'$
    **end if**
    $t = t + 1$
  **until** $T(t) < \tau_0$
  **return** SRU($\mathcal{T}, \Delta, \epsilon$)

Figure 5: Simulated annealing for 2TLIMIDs.

In this paper, we have chosen the acceptance function $P(a(\Delta') = \text{yes} \mid eu, eu', t)$ equal to 1 if $eu' > eu$ and equal to $\exp(\frac{eu' - eu}{T(t)})$ otherwise, where $a(\Delta')$ stands for the acceptance of the proposed strategy $\Delta'$, $eu' = \mathrm{E}^{\epsilon}_{\Delta'}(\mathcal{U})$, $eu = \mathrm{E}^{\epsilon}_{\Delta}(\mathcal{U})$ for the current strategy $\Delta$, and $T$ is an *annealing schedule* that is defined as $T(t+1) = \alpha \cdot T(t)$ where $T(0) = \beta$ with $\alpha < 1$ and $\beta > 0$.

With respect to strategy finding in DLIMIDs, we propose the simulated annealing scheme as shown in Fig. 5, where $\theta$ is repeatedly chosen uniformly at random between 0 and 1. Note that after the annealing phase we apply SRU in order to greedily find a local maximum solution.

## 4 Experimental Results

We have compared the solutions found by SRU and SA to our running example based on twenty randomly chosen initial strategies. Note that SPU was not feasible due to the large number of policies per decision variable. We have chosen a discounting factor $\gamma = 0.95$ and a stopping criterion $\epsilon = 0.01$. After some initial experiments we have chosen $\alpha = 0.995$, $\beta = 0.5$ and $tMin = 3.33 \cdot 10^{-3}$ for the SA parameters. In order to reduce computational load, we assume that the parameters for $P(T^0 \mid M^0)$ and $P(T^t \mid M^t)$ are tied such that we need only estimate three different policies for $P(L^0)$, $P(L^t \mid M^{t-1})$ and $P(T^0 \mid M^0) = P(T^t \mid M^t)$.

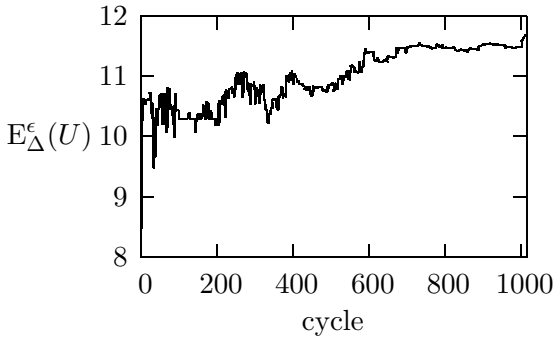



Figure 6: Change in $\mathrm{E}^{\epsilon}_{\Delta}(U)$ for one run of SA. The sudden increase at the end of the run is caused by the subsequent application of SRU.

For the twenty experiments, SRU found strategies with an average expected utility of $\mathrm{E}^{\epsilon}_{\Delta}(\mathcal{U}) = 11.23$ with $\sigma = 0.43$. This required the evaluation of 720 different strategies on average. SA found strategies with an average expected utility of $\mathrm{E}^{\epsilon}_{\Delta}(\mathcal{U}) = 11.51$ with $\sigma = 0.14$. This required the evaluation of 1546 different strategies on average. In 16 out of 20 experiments, SA found strategies with higher expected utility than SRU. Furthermore, due to the random behavior of the algorithm it avoids the local maximum policies found by SRU. For instance, SRU finds a strategy with expected utility 10.29 three out of twenty times, which is equal to the expected utility of the proposed strategy to always treat and never test. The best strategy was found by simulated annealing and has an expected utility of 11.67. The subsequent values of $\mathrm{E}^{\epsilon}_{\Delta}(\mathcal{U})$, found during that experiment, are shown in Fig. 6.

The found strategy can be represented by a *policy graph* (Meuleau et al., 1999); a finite state controller that depicts state transitions, where states represent actions and transitions are induced by observations. Figure 7 depicts the policy graph for the best found strategy. Starting at the left arrow, each time slice constitutes a test decision (circle) and a treatment decision (double circle). Shaded circles stand for positive decisions (i.e., $L^t = yes$ and $T^t = yes$) and clear circles stand for negative decisions (i.e., $L^t = no$ and $T^t = no$). If a test has two outgo-

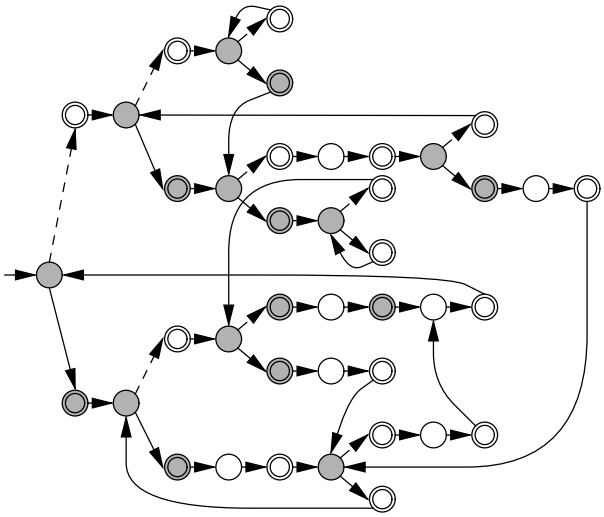


Figure 7: Policy graph for the best strategy found by simulated annealing.

ing arcs, then these stand for a negative finding (dashed arc) and positive finding (solid arc) respectively. Most of the time, the policy graph associates a negative test result with no treatment and a positive test result with treatment.

## 5 Conclusion

In this paper we have demonstrated that reasonable strategies can be found for infinite-horizon DLIMIDs by means of SRU. Although computationally more expensive, SA considerably improves the found strategies by avoiding local maxima. Both SRU and SA do not suffer from the intractability of SPU when the number of informational predecessors increases. The approach does require that good strategies can be found using a limited amount of memory, since otherwise, found strategies will fail to approximate the optimal strategy. This requirement should hold especially between time-slices, since the state-space of memory variables can become prohibitively large when a large part of the observed history is required for optimal decision-making. Although this restricts the types of decision problems that can be managed, DLIMIDs, constructed from a 2TLIMID, allow the representation of large, or even infinite-horizon decision problems, something which standard influence diagrams cannot manage in principle. Hence, 2TLIMIDs are particularly useful in the case of problems that cannot be properly approximated by a short number of time slices.

## References


G.F. Cooper. 1988. A method for using belief networks as influence diagrams. In *Proceedings of the 4th Workshop on Uncertainty in AI*, pages 55–63, University of Minnesota, Minneapolis.

R. Cowell, A. P. Dawid, S. L. Lauritzen, and D. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer.

T. Dean and K. Kanazawa. 1989. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150.

R.A. Howard and J.E. Matheson. 1984. Influence diagrams. In R.A. Howard and J.E. Matheson, editors, *Readings in the Principles and Applications of Decision Analysis*. Strategic Decisions Group, Menlo Park, CA.

S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220:671–680.

S.L. Lauritzen and D. Nilsson. 2001. Representing and solving decision problems with limited information. *Management Science*, 47(9):1235–1251.

N. Meuleau, K.-E. Kim, L.P. Kaelbling, and A.R. Cassandra. 1999. Solving POMDPs by searching the space of finite policies. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 417–426, Stockholm, Sweden.

K.P. Murphy. 2002. *Dynamic Bayesian Networks*. Ph.D. thesis, UC Berkely.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 2 edition.

S. Ross. 1983. *Introduction to Stochastic Dynamic Programming*. Academic Press, New York.

M.A.J. van Gerven, F.J. Díez, B.G. Taal, and P.J.F. Lucas. 2006. Prognosis of high-grade carcinoid tumor patients using dynamic limited-memory influence diagrams. In *Intelligent Data Analysis in bioMedicine and Pharmacology (IDAMAP) 2006*. Accepted for publication.

S.D. Whitehead and D.H. Ballard. 1991. Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83.

# Sensitivity analysis of extreme inaccuracies in Gaussian Bayesian Networks

Miguel A. Gómez-Villegas and Paloma Maín
ma_gv@mat.ucm.es, pmain@mat.ucm.es
Departamento de Estadstica e Investigacin Operativa
Universidad Complutense de Madrid
Plaza de Ciencias 3,
28040 Madrid, Spain

Rosario Susi
rsusi@estad.ucm.es
Departamento de Estadstica e Investigacin Operativa III
Universidad Complutense de Madrid
Avda. Puerta de Hierro s/n,
28040 Madrid, Spain

**Abstract**

We present the behavior of a sensitivity measure defined to evaluate the impact of model inaccuracies over the posterior marginal density of the variable of interest, after the evidence propagation is executed, for extreme perturbations of parameters in Gaussian Bayesian networks. This sensitivity measure is based on the Kullback-Leibler divergence and yields different expressions depending on the type of parameter (mean, variance or covariance) to be perturbed. This analysis is useful to know the extreme effect of uncertainty about some of the initial parameters of the model in a Gaussian Bayesian network. These concepts and methods are illustrated with some examples.

**Keywords**: Gaussian Bayesian Network, Sensitivity analysis, Kullback-Leibler divergence.

## 1 Introduction

Bayesian network is a graphical probabilistic model that provides a graphical framework for complex domains with lots of inter-related variables. Among other authors, Bayesian networks have been studied, by Pearl (1988), Lauritzen (1996), Heckerman (1998) and Jensen (2001).
A sensitivity analysis in a Bayesian network is necessary to study how sensitive is the network's output to inaccuracies or imprecisions in the parameters of the initial network, and therefore to evaluate the network robustness.
In recent years, some sensitivity analysis techniques for Bayesian networks have been developed. In Discrete Bayesian networks Laskey (1995) presents a sensitivity analysis based on computing the partial derivative of a posterior marginal probability with respect to a given parameter, Coupé, et al. (2002) develop an efficient sensitivity analysis based on inference algorithms and Chan, et al. (2005) introduce a sensitivity analysis based on a distance measure. In Gaussian Bayesian networks Castillo, et al. (2003) present a sensitivity analysis based on symbolic propagation and Gómez-Villegas, et al. (2006) develop a sensitivity measure, based on the Kullback-Leibler divergence, to perform the sensitivity analysis.
In this paper, we study the behavior of the sensitivity measure presented by Gómez-Villegas, et al. (2006) for extreme inaccuracies (perturbations) of parameters that describe the Gaussian Bayesian network. To prove that this is a well-

defined measure we are interested in studying the sensitivity measure when one of the parameters is different from the original value. Moreover, we want to proof that if the value of one parameter is similar to the real value then the sensitivity measure is close to zero.
The paper is organized as follows. In Section 2 we briefly introduce definitions of Bayesian networks and Gaussian Bayesian networks, review how propagation in Gaussian Bayesian networks can be performed, and present our working example. In Section 3, we present the sensitivity measure and develop the sensitivity analysis proposed. In Section 4, we obtain the limits of the sensitivity measure for extreme perturbations of the parameters giving the behavior of the measure in the limit so as their interpretation. In Section 5, we perform the sensitivity analysis with the working example for some extreme imprecisions. Finally, the paper ends with some conclusions.

## 2 Gaussian Bayesian Networks and Evidence propagation

**Definition 1** (Bayesian network)**.** A Bayesian network is a pair ($G$,$P$) where $G$ is a directed acyclic graph (DAG), which nodes corresponding to random variables $\mathbf{X}$=$\{X_1, ..., X_n\}$ and edges representing probabilistic dependencies, and $P$=$\{p(x_1|pa(x_1)), ..., p(x_n|pa(x_n))\}$ is a set of conditional probability densities (one for each variable) where $pa(x_i)$ is the set of parents of node $X_i$ in $G$. The set $P$ defines the associated joint probability density as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|pa(x_i)). \qquad (1)$$

As usual we work with a variable of interest, so the network's output is the information about this variable of interest after the evidence propagation.

**Definition 2** (Gaussian Bayesian network)**.** A Gaussian Bayesian network is a Bayesian network over $\mathbf{X}$=$\{X_1, ..., X_n\}$ with a multivariate normal distribution $N(\mu, \Sigma)$, then the joint density is given by $f(\mathbf{x}) = (2\pi)^{-n/2}|\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)\right\}$ where $\mu$ is the $n$-dimensional mean vector, $\Sigma$ $n\times n$ the covariance matrix and $|\Sigma|$ the determinant of $\Sigma$.

The conditional density associated with $X_i$ for $i = 1, ..., n$ in equation (1), is the univariate normal distribution, with density $f(x_i|pa(x_i)) \sim N\left(\mu_i + \sum_{j=1}^{i-1} \beta_{ij}(x_j - \mu_j), \nu_i\right)$ where $\beta_{ij}$ is the regression coefficient of $X_j$ in the regression of $X_i$ on the parents of $X_i$, and $\nu_i = \Sigma_{ii} - \Sigma_{iPa(x_i)}\Sigma^{-1}_{Pa(xi)}\Sigma^{'}_{iPa(x_i)}$ is the conditional variance of $X_i$ given its parents.

Different algorithms have been proposed to propagate the evidence of some nodes in Gaussian Bayesian networks. We present an incremental method, updating one evidential variable at a time (see Castillo, et al. 1997) based on computing the conditional probability density of a multivariate normal distribution given the evidential variable $X_e$.
For the partition $\mathbf{X} = (\mathbf{Y}, E)$, with $\mathbf{Y}$ the set of non-evidential variables, where $X_i \in \mathbf{Y}$ is the variable of interest, and $E$ is the evidence variable, then, the conditional probability distribution of $\mathbf{Y}$, given the evidence $E = \{X_e = e\}$, is multivariate normal with parameters

$$\mu^{\mathbf{Y}|E=e} = \mu_{\mathbf{Y}} + \Sigma_{\mathbf{Y}E}\Sigma^{-1}_{EE}(e - \mu_E)$$

$$\Sigma^{\mathbf{Y}|E=e} = \Sigma_{\mathbf{YY}} - \Sigma_{\mathbf{Y}E}\Sigma^{-1}_{EE}\Sigma_{E\mathbf{Y}}$$

Therefore, the variable of interest $X_i \in \mathbf{Y}$ after the evidence propagation is

$$X_i|E = e \sim N(\mu_i^{\mathbf{Y}|E=e}, \sigma_{ii}^{\mathbf{Y}|E=e}) \equiv$$

$$\equiv N\left(\mu_i + \frac{\sigma_{ie}}{\sigma_{ee}}(e - \mu_e), \sigma_{ii} - \frac{\sigma_{ie}^2}{\sigma_{ee}}\right) \qquad (2)$$

where $\mu_i$ and $\mu_e$ are the means of $X_i$ and $X_e$ respectively before the propagation, $\sigma_{ii}$ and $\sigma_{ee}$ the variances of $X_i$ and $X_e$ respectively before propagating the evidence, and $\sigma_{ie}$ the covariance between $X_i$ and $X_e$ before the evidence propagation.

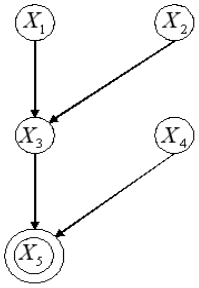



Figure 1: DAG of the Gaussian Bayesian network

We illustrate the concept of a Gaussian Bayesian network and the evidence propagation method with next example.

**Example 1.** Assume that we are interested in how a machine works. This machine is made up of 5 elements, the variables of the problem, connected as the network in Figure 1. Let us consider the time each element is working has a normal distribution and we are interested in the last element of the machine ($X_5$).

Being $\mathbf{X} = \{X_1, X_2, X_3, X_4, X_5\}$ normally distributed, $\mathbf{X} \sim N(\mu, \Sigma)$, with parameters

$$\mu = \begin{pmatrix} 2 \\ 3 \\ 3 \\ 4 \\ 5 \end{pmatrix}; \Sigma = \begin{pmatrix} 3 & 0 & 6 & 0 & 6 \\ 0 & 2 & 2 & 0 & 2 \\ 6 & 2 & 15 & 0 & 15 \\ 0 & 0 & 0 & 2 & 4 \\ 6 & 2 & 15 & 4 & 26 \end{pmatrix}$$

Considering the evidence $E = \{X_2 = 4\}$, after evidence propagation we obtain that $X_5|X_2 = 4 \sim N(6, 24)$ and the joint distribution is normal with parameters

$$\mu^{\mathbf{Y}|X_2=4} = \begin{pmatrix} 2 \\ 4 \\ 4 \\ 6 \end{pmatrix};$$

$$\Sigma^{\mathbf{Y}|X_2=4} = \begin{pmatrix} 3 & 6 & 0 & 6 \\ 6 & 13 & 0 & 13 \\ 0 & 0 & 2 & 4 \\ 6 & 13 & 4 & 24 \end{pmatrix}$$

## 3 Sensitivity Analysis and non-influential parameters

The Sensitivity Analysis proposed is as follows: Let us suppose the model before propagating the evidence is $N(\mu, \Sigma)$ with one evidential variable $X_e$, whose value is known. After propagating this evidence we obtain the marginal density of interest $f(x_i|e)$. Next, we add a *perturbation* $\delta$ to one of the parameters in the model before propagating the evidence (this parameter is supposed to be inaccurate thus $\delta$ reflects this inaccuracy) and perform the evidence propagation, to get $f(x_i|e, \delta)$. In some cases, the perturbation $\delta$ has some restrictions to get admissible parameters.

The effect of adding the perturbation is computed by comparing those density functions by means of the sensitivity measure. That measure is based on the Kullback-Leibler divergence between the target marginal density obtained after evidence propagation, considering the model with and without the perturbation

**Definition 3** (Sensitivity measure)**.** Let $(G,P)$ be a Gaussian Bayesian network $N(\mu, \Sigma)$. Let $f(x_i|e)$ be the marginal density of interest after evidence propagation and $f(x_i|e, \delta)$ the same density when the perturbation $\delta$ is added to one parameter of the initial model. Then, the sensitivity measure is defined by

$$S^{p_j}(f(x_i|e), f(x_i|e, \delta)) =$$

$$= \int_{-\infty}^{\infty} f(x_i|e) \ln \frac{f(x_i|e)}{f(x_i|e, \delta)} dx_i \qquad (3)$$

where the subscript $p_j$ is the inaccurate parameter and $\delta$ the proposed perturbation, being the new value of the parameter $p_j^\delta = p_j + \delta$.

For small values of the sensitivity measure we can conclude our Bayesian network is robust for that perturbation.

### 3.1 Mean vector inaccuracy

Three different situations are possible depending on the element of $\mu$ to be perturbed, i.e., the perturbation can affect the mean of the variable of interest $X_i \in \mathbf{Y}$, the mean of the

evidence variable $X_e \in E$, or the mean of any other variable $X_j \in \mathbf{Y}$ with $j \neq i$. Developing the sensitivity measure (3), we obtain next propositions,

**Proposition 1.** *For the perturbation $\delta \in \Re$ in the mean vector $\mu$, the sensitivity measure is as follows*

• *When the perturbation is added to the mean of the variable of interest, $\mu_i^\delta = \mu_i + \delta$, and $X_i|E = e, \delta \sim N(\mu_i^{\mathbf{Y}|E=e} + \delta, \sigma_{ii}^{\mathbf{Y}|E=e})$,*

$$S^{\mu_i}(f(x_i|e), f(x_i|e,\delta)) = \frac{\delta^2}{2\sigma_{ii}^{\mathbf{Y}|E=e}}.$$

• *If the perturbation is added to the mean of the evidential variable, $\mu_e^\delta = \mu_e + \delta$, the posterior density of the variable of interest, with the perturbation, is $X_i|E = e, \delta \sim N(\mu_i^{\mathbf{Y}|E=e} - \frac{\sigma_{ie}}{\sigma_{ee}}\delta, \sigma_{ii}^{\mathbf{Y}|E=e})$, then*

$$S^{\mu_e}(f(x_i|e), f(x_i|e,\delta)) = \frac{\delta^2}{2\sigma_{ii}^{\mathbf{Y}|E=e}} \left(\frac{\sigma_{ie}}{\sigma_{ee}}\right)^2.$$

• *The perturbation $\delta$ added to the mean of any other non-evidential variable, different from the variable of interest, has no influence over $X_i$, then $f(x_i|e,\delta) = f(x_i|e)$, and the sensitivity measure is zero.*

### 3.2 Covariance matrix inaccuracy

There are six possible different situations, depending on the parameter of the covariance matrix $\Sigma$ to be changed; three if the perturbation is added to the variances (elements in the diagonal of $\Sigma$) and other three if the perturbation is added to the covariances of $\Sigma$. When the covariance matrix is perturbed, the structure of the network can change. Those changes are presented in the precision matrix of the perturbed network, that is, the inverse of the covariance matrix with perturbation $\delta$. To guarantee the normality of the network it is necessary $\Sigma^{\mathbf{Y}|E=e,\delta}$ to be a positive definite matrix in all cases presented in next proposition

**Proposition 2.** *Adding the perturbation $\delta \in \Re$ to the covariance matrix $\Sigma$, the sensitivity measure obtained is*

• *If the perturbation is added to the variance of the variable of interest, being $\sigma_{ii}^\delta = \sigma_{ii} + \delta \quad with \quad \delta > -\sigma_{ii} + \frac{\sigma_{ie}^2}{\sigma_{ee}}$, then $X_i|E = e, \delta \sim N(\mu_i^{\mathbf{Y}|E=e}, \sigma_{ii}^{\mathbf{Y}|E=e,\delta})$ where $\sigma_{ii}^{\mathbf{Y}|E=e,\delta} = \sigma_{ii} + \delta - \frac{\sigma_{ie}^2}{\sigma_{ee}}$ and $S^{\sigma_{ii}}(f(x_i|e), f(x_i|e,\delta)) =$*

$$= \frac{1}{2}\left[\ln\left(1 + \frac{\delta}{\sigma_{ii}^{\mathbf{Y}|E=e}}\right) - \frac{\delta}{\sigma_{ii}^{\mathbf{Y}|E=e,\delta}}\right].$$

• *When the perturbation is added to the variance of the evidential variable, being $\sigma_{ee}^\delta = \sigma_{ee} + \delta \quad and \quad \delta > -\sigma_{ee}(1 - \max_{X_j \in \mathbf{Y}} \rho_{je}^2)$ with $\rho_{je}$ the corresponding correlation coefficient, the posterior density of interest is $X_i|E = e, \delta \sim N(\mu_i^{\mathbf{Y}|E=e,\delta}, \sigma_{ii}^{\mathbf{Y}|E=e,\delta})$ with $\mu_i^{\mathbf{Y}|E=e,\delta} = \mu_i + \frac{\sigma_{ie}}{\sigma_{ee}+\delta}(e - \mu_e)$ and $\sigma_{ii}^{\mathbf{Y}|E=e,\delta} = \sigma_{ii} - \frac{\sigma_{ie}^2}{\sigma_{ee}+\delta}$ therefore*

$$S^{\sigma_{ee}}(f(x_i|e), f(x_i|e,\delta)) = \frac{1}{2}\left[\ln\left(\frac{\sigma_{ii}^{\mathbf{Y}|E=e,\delta}}{\sigma_{ii}^{\mathbf{Y}|E=e}}\right) + \right.$$
$$\left. + \frac{\frac{\sigma_{ie}^2}{\sigma_{ee}}\left(\frac{-\delta}{\sigma_{ee}+\delta}\right)\left(1 + (e-\mu_e)^2\left(\frac{-\delta}{(\sigma_{ee}+\delta)\sigma_{ee}}\right)\right)}{\sigma_{ii}^{\mathbf{Y}|E=e,\delta}}\right].$$

• *The perturbation $\delta$ added to the variance of any other non-evidential variable $X_j \in \mathbf{Y}$ with $j \neq i$, $\sigma_{jj}^\delta = \sigma_{jj} + \delta$, has no influence over $X_i$, therefore $f(x_i|e,\delta) = f(x_i|e)$ and the sensitivity measure is zero.*

• *When the covariance between the variable of interest $X_i$ and the evidential variable $X_e$ is perturbed, $\sigma_{ie}^\delta = \sigma_{ie} + \delta = \sigma_{ei}^\delta$ and $-\sigma_{ie} - \sqrt{\sigma_{ii}\sigma_{ee}} < \delta < -\sigma_{ie} + \sqrt{\sigma_{ii}\sigma_{ee}}$, then $X_i|E = e, \delta \sim N(\mu_i^{\mathbf{Y}|E=e,\delta}, \sigma_{ii}^{\mathbf{Y}|E=e,\delta})$ with $\mu_i^{\mathbf{Y}|E=e,\delta} = \mu_i + \frac{(\sigma_{ie}+\delta)}{\sigma_{ee}}(e - \mu_e)$ and $\sigma_{ii}^{\mathbf{Y}|E=e,\delta} = \sigma_{ii} - \frac{(\sigma_{ie}+\delta)^2}{\sigma_{ee}}$ the sensitivity measure obtained is*

$S^{\sigma_{ie}}(f(x_i|e), f(x_i|e,\delta)) =$

$$= \frac{1}{2}\left[\ln\left(1 - \frac{\delta^2 + 2\sigma_{ie}\delta}{\sigma_{ee}\sigma_{ii}^{\mathbf{Y}|E=e}}\right) + \right.$$

$$\left. + \frac{\sigma_{ii}^{\mathbf{Y}|E=e} + \left(\frac{\delta}{\sigma_{ee}}(e-\mu_e)\right)^2}{\sigma_{ii}^{\mathbf{Y}|E=e,\delta}} - 1\right].$$

• *If we add the perturbation to any other covariance, i.e., between the variable of interest $X_i$ and any other non-evidential variable $X_j$ or between the evidence variable $X_e$ and $X_j \in \mathbf{Y}$ with $j \neq i$, the posterior probability density of the variable of interest $X_i$ is the same as without perturbation and therefore the sensitivity measure is zero.*

## 4 Extreme behavior of the Sensitivity Measure

When using the sensitivity measure, that describes how sensitive is the variable of interest when a perturbation is added to a inaccurate parameter, it would be interesting to know how is the sensitivity measure when the perturbation $\delta \in \Re$ is extreme. Then, we study the behavior of that measure for extreme perturbations so as the limit of the sensitivity measure.
Next propositions present the results about the limits of the sensitivity measures in all cases given in Propositions 1 and 2,

**Proposition 3.** *When the perturbation added to the mean vector is extreme, the sensitivity measure is as follows,*

1. • $\lim_{\delta \longrightarrow \pm\infty} S^{\mu_i}(f(x_i|e), f(x_i|e,\delta)) = \infty$
   • $\lim_{\delta \longrightarrow 0} S^{\mu_i}(f(x_i|e), f(x_i|e,\delta)) = 0$

2. • $\lim_{\delta \longrightarrow \pm\infty} S^{\mu_e}(f(x_i|e), f(x_i|e,\delta)) = \infty$
   • $\lim_{\delta \longrightarrow 0} S^{\mu_e}(f(x_i|e), f(x_i|e,\delta)) = 0.$

*Proof.* It follows directly. □

**Proposition 4.** *When the extreme perturbation is added to the elements of the covariance matrix and the correlation coefficient of $X_i$ and $X_e$ is $0 < \rho_{ie}^2 < 1$, the results are,*

1. • $\lim_{\delta \longrightarrow \infty} S^{\sigma_{ii}}(f(x_i|e), f(x_i|e,\delta)) = \infty$ *but* $S^{\sigma_{ii}}(f(x_i|e), f(x_i|e,\delta)) = o(\delta)$

   • $\lim_{\delta \longrightarrow M_{ii}} S^{\sigma_{ii}}(f(x_i|e), f(x_i|e,\delta)) = \infty$ *with* $M_{ii} = -\sigma_{ii} + \frac{\sigma_{ie}^2}{\sigma_{ee}} = -\sigma_{ii}(1-\rho_{ie}^2)$

   • $\lim_{\delta \longrightarrow 0} S^{\sigma_{ii}}(f(x_i|e), f(x_i|e,\delta)) = 0$

2. • $\lim_{\delta \longrightarrow \infty} S^{\sigma_{ee}}(f(x_i|e), f(x_i|e,\delta)) =$
   $= \frac{1}{2}\left[-ln(1-\rho_{ie}^2) - \rho_{ie}^2\left(1 - \frac{(e-\mu_e)^2}{\sigma_{ee}}\right)\right]$

   • $\lim_{\delta \longrightarrow M_{ee}} S^{\sigma_{ee}}(f(x_i|e), f(x_i|e,\delta)) =$
   $= \frac{1}{2}\left[ln\left(\frac{M_{ee}^* - \rho_{ie}^2}{M_{ee}^*(1-\rho_{ie}^2)}\right) + \frac{\rho_{ie}^2(1-M_{ee}^*)}{M_{ee}^* - \rho_{ie}^2}\right.$
   $\left.\left(1 + \frac{(e-\mu_e)^2}{\sigma_{ee}}\left(\frac{1-M_{ee}^*}{M_{ee}^*}\right)\right)\right]$
   *where* $M_{ee} = -\sigma_{ee}(1 - M_{ee}^*)$ *being* $M_{ee}^* = \max_{X_j} \rho_{je}^2$

   • $\lim_{\delta \longrightarrow 0} S^{\sigma_{ee}}(f(x_i|e), f(x_i|e,\delta)) = 0$

3. • $\lim_{\delta \longrightarrow M_{ie}^1} S^{\sigma_{ie}}(f(x_i|e), f(x_i|e,\delta)) = \infty$

   • $\lim_{\delta \longrightarrow M_{ie}^2} S^{\sigma_{ie}}(f(x_i|e), f(x_i|e,\delta)) = \infty$
   *with* $M_{ie}^1 = -\sigma_{ie} - \sqrt{\sigma_{ii}\sigma_{ee}}$ *and* $M_{ie}^2 = -\sigma_{ie} + \sqrt{\sigma_{ii}\sigma_{ee}}$

   • $\lim_{\delta \longrightarrow 0} S^{\sigma_{ie}}(f(x_i|e), f(x_i|e,\delta)) = 0.$

*Proof.* 1. • It follows directly.

• When $\sigma_{ii}^{\delta} = \sigma_{ii} + \delta$ the new variance of $X_i$ is $\sigma_{ii}^{\mathbf{Y}|E=e,\delta} = \sigma_{ii}^{\mathbf{Y}|E=e} + \delta$.
Being $\sigma_{ii}^{\mathbf{Y}|E=e,\delta} > 0$ then $\delta > -\sigma_{ii}^{\mathbf{Y}|E=e}$.
Naming $M_{ii} = -\sigma_{ii}^{\mathbf{Y}|E=e}$ and considering $x = \sigma_{ii}^{\mathbf{Y}|E=e} + \delta$ we have

$\lim_{\delta \longrightarrow M_{ii}} S^{\sigma_{ii}}(f(x_i|e), f(x_i|e,\delta)) =$
$= \lim_{x \longrightarrow 0} \frac{1}{2x} \left[ x \ln(x) - x \ln(\sigma_{ii}^{\mathbf{Y}|E=e}) - x + \right.$
$\left. + \sigma_{ii}^{\mathbf{Y}|E=e} \right] = \infty.$

• It follows directly.

2. • $\lim_{\delta \longrightarrow \infty} S^{\sigma_{ee}}(f(x_i|e), f(x_i|e,\delta)) =$
$= \frac{1}{2} \left[ \ln \left( \frac{\sigma_{ii}}{\sigma_{ii}^{\mathbf{Y}|E=e}} \right) + \frac{\frac{-\sigma_{ie}^2}{\sigma_{ee}} \left( 1 - \frac{(e-\mu_e)^2}{\sigma_{ee}} \right)}{\sigma_{ii}} \right]$
with $\sigma_{ii}^{\mathbf{Y}|E=e} = \sigma_{ii}(1-\rho_{ie}^2)$ and $\rho_{ie}^2 = \frac{\sigma_{ie}^2}{\sigma_{ii}\sigma_{ee}}$ the limit is
$= \frac{1}{2} \left[ -ln(1-\rho_{ie}^2) - \rho_{ie}^2 \left( 1 - \frac{(e-\mu_e)^2}{\sigma_{ee}} \right) \right].$

• When $\sigma_{ee}^{\delta} = \sigma_{ee} + \delta$, the new conditional variance for all non evidential variables is $\sigma_{jj}^{\mathbf{Y}|E=e,\delta} = \sigma_{jj} - \frac{\sigma_{je}^2}{\sigma_{ee}+\delta}$ for all $X_j \in \mathbf{Y}$.
If we impose $\sigma_{jj}^{\mathbf{Y}|E=e,\delta} > 0$ for all $X_j \in \mathbf{Y}$ then $\delta$ must satisfy next condition $\delta > -\sigma_{ee}(1 - \max_{X_j \in \mathbf{Y}} \rho_{je}^2)$.
Naming $M_{ee}^* = \max_{X_j} \rho_{je}^2$ and $M_{ee} = -\sigma_{ee}(1 - M_{ee}^*)$ then we have
$\lim_{\delta \longrightarrow M_{ee}} S^{\sigma_{ee}}(f(x_i|e), f(x_i|e,\delta)) =$
$= \lim_{\delta \longrightarrow M_{ee}} \frac{1}{2} \left[ \ln \left( \frac{\sigma_{ii} - \frac{\sigma_{ie}^2}{\sigma_{ee}+\delta}}{\sigma_{ii} - \frac{\sigma_{ie}^2}{\sigma_{ee}}} \right) + \right.$
$\left. \frac{\frac{\sigma_{ie}^2}{\sigma_{ee}} \left( \frac{-\delta}{\sigma_{ee}+\delta} \right) \left( 1 + (e-\mu_e)^2 \left( \frac{-\delta}{(\sigma_{ee}+\delta)\sigma_{ee}} \right) \right)}{\sigma_{ii} - \frac{\sigma_{ie}^2}{\sigma_{ee}+\delta}} \right]$
with $\rho_{ie}^2 = \frac{\sigma_{ie}^2}{\sigma_{ii}\sigma_{ee}}$
$= \frac{1}{2} \left[ \ln \left( \frac{\sigma_{ii}\sigma_{ee}M_{ee}^* - \sigma_{ie}^2}{M_{ee}^*(\sigma_{ii}\sigma_{ee} - \sigma_{ie}^2)} \right) + \right.$
$\left. + \frac{\frac{\sigma_{ie}^2}{\sigma_{ee}} \left( \frac{1-M_{ee}^*}{M_{ee}^*} \right) \left( 1 + \frac{(e-\mu_e)^2}{\sigma_{ee}} \left( \frac{1-M_{ee}^*}{M_{ee}^*} \right) \right)}{M_{ee}^* - \rho_{ie}^2} \right] =$
$= \frac{1}{2} \left[ ln \left( \frac{M_{ee}^* - \rho_{ie}^2}{M_{ee}^*(1-\rho_{ie}^2)} \right) + \frac{\rho_{ie}^2(1-M_{ee}^*)}{M_{ee}^* - \rho_{ie}^2} \right.$
$\left. \left( 1 + \frac{(e-\mu_e)^2}{\sigma_{ee}} \left( \frac{1-M_{ee}^*}{M_{ee}^*} \right) \right) \right].$

• It follows directly.

3. • If we make $\sigma_{ie}^{\delta} = \sigma_{ie} + \delta$, the new conditional variance is $\sigma_{ii}^{\mathbf{Y}|E=e,\delta} = \sigma_{ii}^{\mathbf{Y}|E=e} - \frac{\delta^2 + 2\delta\sigma_{ie}}{\sigma_{ee}}$.
Then, if we impose $\sigma_{ii}^{\mathbf{Y}|E=e,\delta} > 0$, $\delta$ must satisfy the next condition $-\sigma_{ie} - \sqrt{\sigma_{ii}\sigma_{ee}} < \delta < -\sigma_{ie} + \sqrt{\sigma_{ii}\sigma_{ee}}$.
First, naming $M_{ie}^2 = -\sigma_{ie} + \sqrt{\sigma_{ii}\sigma_{ee}}$, we calculate $\lim_{\delta \longrightarrow M_{ie}^2} S^{\sigma_{ie}}(f(x_i \mid e), f(x_i \mid e, \delta))$.
But $\delta \rightarrow M_{ie}^2$ is equivalent to $(\delta^2 + 2\delta\sigma_{ie}) \rightarrow \sigma_{ee}\sigma_{ii}^{\mathbf{Y}|E=e}$ and given that
$S^{\sigma_{ie}}(f(x_i \mid e), f(x_i \mid e, \delta)) =$
$= \frac{1}{2} \left[ \ln \left( \frac{\sigma_{ee}\sigma_{ii}^{\mathbf{Y}|E=e} - (\delta^2 + 2\delta\sigma_{ie})}{\sigma_{ee}\sigma_{ii}^{\mathbf{Y}|E=e}} \right) + \right.$
$\left. + \frac{\sigma_{ee}\sigma_{ii}^{\mathbf{Y}|E=e} + \left( \frac{\delta}{\sigma_{ee}}(e-\mu_e) \right)^2}{\sigma_{ee}\sigma_{ii}^{\mathbf{Y}|E=e} - (\delta^2 + 2\delta\sigma_{ie})} - 1 \right]$
and as $\lim_{x \longrightarrow 0} \left[ \ln x + \frac{k}{x} \right] = \infty$ for every $k$, then we get $\lim_{\delta \longrightarrow M_{ie}^2} S^{\sigma_{ie}}(f(x_i \mid e), f(x_i \mid e, \delta)) = \infty$.

• Analogously, the other limit is also $\lim_{\delta \longrightarrow M_{ie}^1} S^{\sigma_{ie}}(f(x_i \mid e), f(x_i \mid e, \delta)) = \infty$.

• It follows directly.

□

The behavior of the sensitivity measure is the expected one, except when the extreme perturbation is added to the evidential variance, because with known evidence, the posterior density of interest with the perturbation in the model $f(x_i|e,\delta)$ is not so different of the posterior density of interest without the perturbation $f(x_i|e)$, therefore although an extreme perturbation added to the evidential variance can exist, the sensitivity measure tends to a finite value.

## 5 Experimental results

**Example 2.** Consider the Gaussian Bayesian network given in Example 1. Experts disagree with definition of the variable of interest $X_5$, then the mean could be $\mu_5^{\delta_1} = 2 = \mu_5 + \delta_1$ (with $\delta_1 = -3$), the variance could be $\sigma_{55}^{\delta_2} = 24$ with $\delta_2 = -2$ and the covariances between $X_5$ and evidential variable $X_2$ could be $\sigma_{52}^{\delta_3} = 3$ with $\delta_3 = 1$ (the same to $\sigma_{25}$); with the variance of the evidential variable being $\sigma_{22}^{\delta_4} = 4$ with $\delta_4 = 2$ and between $X_5$ and other non-evidential variable $X_3$ that could be $\sigma_{53}^{\delta_5} = 13$ with $\delta_5 = -2$ (the same to $\sigma_{35}$); moreover, there are different opinions about $X_3$, because they suppose that $\mu_3$ could be $\mu_3^{\delta_6} = 7$ with $\delta_6 = 4$, that $\sigma_{33}$ could be $\sigma_{33}^{\delta_7} = 17$ with $\delta_7 = 2$, and that $\sigma_{32}$ could be $\sigma_{32}^{\delta_8} = 1$ with $\delta_8 = -1$ (the same to $\sigma_{23}$).

The sensitivity measure for those inaccuracy parameters yields
$S^{\mu_5}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_1)) = 0.1875$
$S^{\sigma_{55}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_2)) = 0.00195$
$S^{\sigma_{52}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_3)) = 0.00895$
$S^{\sigma_{22}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_4)) = 0.00541$
$S^{\sigma_{53}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_5)) = 0$
$S^{\mu_3}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_6)) = 0$
$S^{\sigma_{33}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_7)) = 0$
$S^{\sigma_{32}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_8)) = 0$
As these values of the sensitivity measures are small we can conclude that the perturbations presented do not affect the variable of interest and therefore the network can be considered robust. Also, the inaccuracies about the non-evidential variable $X_3$ do not disturb the posterior marginal density of interest, being the sensitivity measure zero in all cases. If experts think that the sensitivity measure obtained for the mean of the variable of interest is large enough then they should review the information about this variable.
Moreover, we have implemented an algorithm (see Appendix 1) to compute all the sensitivity measures that can influence over the variable of interest $X_i$; this algorithm compare those sensitivity measures computed with a threshold $s$ fixed by experts. Then, if the sensitivity measure is larger than the threshold, the parameter should be reviewed.

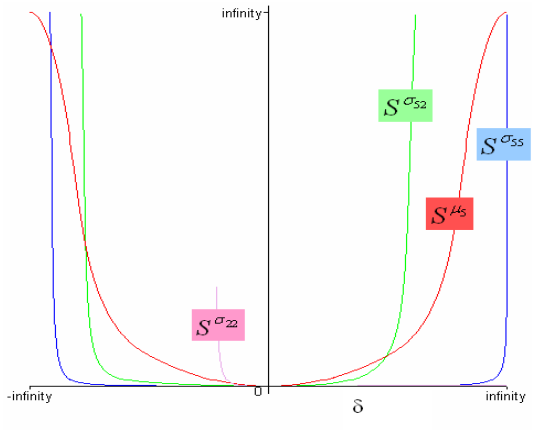



Figure 2: Sensitivity measures obtained in the example for any perturbation value

The extreme behavior of the sensitivity measure for some particular cases, is given as follows When $\delta_1 = 20$, the sensitivity measure is $S^{\mu_5}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_1)) = 8.33$ and with the perturbation $\delta_1 = -25$, $S^{\mu_5}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_1)) = 13.02$.
If the perturbation $\delta_2 = 1000$, $S^{\sigma_{55}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_2)) = 1.39$ therefore as the perturbation increases to infinity the sensitivity measure grows very slowly, in fact $S^{\sigma_{55}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_2)) = o(\delta)$ as stated before. However if $\delta_2 = -23$, the sensitivity measure is $S^{\sigma_{55}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_2)) = 9.91$.
We do not present the sensitivity measure when $\delta_3$ is extreme because $\delta_3$ must be in $(-\sqrt{2}, \sqrt{2})$ to keep the covariance matrix of the network with the perturbation $\delta_3$ positive definite.
Finally, when $\delta_4 = 100$, the sensitivity measure is $S^{\sigma_{22}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_4)) = 0.02$ (where the limit of $S^{\sigma_{22}}$ when $\delta$ tends to infinity is 0.0208) and with $\delta_4 = -1.73$, $S^{\sigma_{22}}(f(x_5|X_2 = 4), f(x_5|X_2 = 4, \delta_4)) = 2.026$ (being the limit when $\delta$ tends to $M_{ee}$ 2.1212).
In Figure 2, we observed the sensitivity measures, considered as a function of $\delta$; the graph shows the behavior of the measure when $\delta \in \Re$.

## 6 Conclusions

In this paper we study the behavior of the sensitivity measure, that compares the marginal density of interest when the model of the Gaussian Bayesian network is described with and without a perturbation $\delta \in \Re$, when the perturbation is extreme. Considering a large perturbation the sensitivity measure is large too except when the extreme perturbation is added to the evidence variance. Therefore, although the evidence variance were large and different from the variance in the original model, the sensitivity measure would be limited by a finite value, that is because the evidence about this variable explains the behavior of the variable of interest regardless its inaccurate variance.

Moreover, in all possible cases of the sensitivity measure, if the perturbation added to a parameter tends to zero, the sensitivity measure is zero too.

The study of the behavior of the sensitivity measure is useful to prove that this is a well-defined measure to develop a sensitivity analysis in Gaussian Bayesian networks even if the proposed perturbation is extreme.

The posterior research is focused on perturbing more than one parameter simultaneously so as with more than one variable of interest.

## References


Castillo, E., Gutirrez, J.M., Hadi, A.S. 1997. *Expert Systems and Probabilistic Network Models.* New York: Springer–Verlag.

Castillo, E., Kjærulff, U. 2003. Sensitivity analysis in Gaussian Bayesian networks using a symbolic-numerical technique. *Reliability Engineering and System Safety*, 79:139–148.

Chan, H., Darwiche, A. 2005. A distance Measure for Bounding Probabilistic Belief Change. *International Journal of Approximate Reasoning*, 38(2):149–174.

Coupé, V.M.H., van der Gaag, L.C. 2002. Propierties of sensitivity analysis of Bayesian belief networks. *Annals of Mathematics and Artificial Intelligence*, 36:323–356.

Gómez-Villegas, M.A., Maín, P., Susi, R. 2006. Sensitivity analysis in Gaussian Bayesian networks using a divergence measure. *Communications in Statistics–Theory and Methods*, accepted for publication.

Heckerman, D. 1998. A tutorial on learning with Bayesian networks. In: Jordan, M. I., eds, *Learning in Graphical Models.* Cambridge, Massachusetts: MIT Press.

Jensen, F.V. 2001. *Bayesian Networks and Decision Graphs.* New York: Springer.

Kullback, S., Leibler, R.A. 1951. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22:79–86.

Laskey, K.B. 1995. Sensitivity Analysis for Probability Assessments in Bayesian Networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(6):901–909.

Lauritzen, S.L. 1996. *Graphical Models.* Oxford: Clarendon Press.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference*, Morgan Kaufmann: Palo Alto.


### Appendix 1

The algorithm that computes the sensitivity measures and determines the parameters in the network is available at the URL:
http://www.ucm.es/info/eue/pagina/
/APOYO/RosarioSusiGarcia/S_algorithm.pdf


### Acknowledgments

This research was supported by the MEC from Spain Grant MTM2005-05462 and Universidad Complutense de Madrid-Comunidad de Madrid Grant UCM2005-910395.

# Learning Bayesian Networks Structure using Markov Networks

Christophe Gonzales and Nicolas Jouve
Laboratoire d'Informatique de Paris VI
8 rue du capitaine Scott 75015 Paris, France

**Abstract**

This paper addresses the problem of learning a Bayes net (BN) structure from a database. We advocate first searching the Markov networks (MNs) space to obtain an initial RB and, then, refining it into an optimal RB. More precisely, it can be shown that under classical assumptions our algorithm obtains the optimal RB moral graph in polynomial time. This MN is thus optimal w.r.t. inference. The process is attractive in that, in addition to providing optimality guarrantees, the MN space is substantially smaller than the traditional search spaces (those of BNs and equivalent classes (ECs)). In practice, we face the classical shortcoming of constraint-based methods, namely the unreliability of high-order conditional independence tests, and handle it using efficient conditioning set computations based on graph triangulations. Our preliminary experimentations are promising, both in terms of the quality of the produced solutions and in terms of time responses.

## 1 Introduction

Effective modeling of uncertainty is essential to most AI applications. For fifteen years probabilistic graphical models like Bayesian nets (BN) and Markov nets (MN), a.k.a. Markov random fields, (Cowell et al., 1999; Jensen, 1996; Pearl, 1988) have proved to be well suited and computationally efficient to deal with uncertainties in many practical applications. Their key feature consists in exploiting probabilistic conditional independences (CIs) to decompose a joint probability distribution over a set of random variables as a product of functions of smaller sets, thus allowing a compact representation of the joint probability as well as efficient inference algorithms (Allen and Darwiche, 2003; Madsen and Jensen, 1999). These independences are encoded in the graphical structure of the model, which is directed for BNs and undirected for MNs. In this paper, we propose an algorithm to learn a BN structure from a data sample of the distribution of interest.

There exist three main classes of algorithms for learning BN from data. The first one tries to determine the set of all probabilistic CIs using statistical independence tests (Verma and Pearl, 1990; Spirtes et al., 2000). Such tests have been criticized in the literature as they can only be applied with small conditioning sets, thus ruling out complex BNs. A more popular approach consists in searching the BN structures space, optimizing some score (BDeu, MDL, etc.) measuring how well the structure fits data (Heckerman et al., 1995; Lam and Bacchus, 1993). Unfortunately, the number of BNs is super-exponential in the number of random variables (Robinson, 1973) and an exhaustive comparison of all the structures is impossible. One way out is to use local search algorithms parsing efficiently the BN structure space, moving from one BN to the next one by performing simple graphical modifications (Heckerman et al., 1995). However, these suffer from the existence of multiple BNs representing the same set of CIs. This not only lengthens the search but may also trap it into local optima. To avoid these problems, the third class of approaches (Munteanu and Bendou, 2001; Chickering, 2002) searches the space of BN equivalence classes (EC). In this space, equivalent BNs are represented by a unique partially directed graph. In addition to speeding-up the search by avoiding useless moves from one BN to an equivalent one, this class of algorithms possesses nice theoretical properties. For instance, under DAG-isormorphism, a classical hypothesis on the data generative distribution, and in the limit of a large database, the GES algorithm is theoretically able to recover the optimal BN structure (Chickering, 2002). This property

is remarkable as very few search algorithms are able to guarantee the quality of the returned structure.

Although, in theory, the EC space seems more attractive than the BN space, it suffers in practice from two problems: i) its neighborhood is exponential in the number of nodes (Chickering, 2002) and ii) the EC space size is roughly equal to that of the BN space (Gillispie and Perlman, 2001). It would thus be interesting to find a space preserving the optimality feature of the EC space exploited by GES while avoiding the above problems. For this purpose, the MN space seems a good candidate as, like the EC space, its equivalence classes are singletons. Moreover, it is exponentially smaller than the BN space and, by its undirected nature, its neighborhood is also exponentially smaller than that of EC. This suggests that searching the MN space instead of the EC space can lead to significant improvements.

To support this claim, we propose a learning algorithm that mainly performs its search in the MN space. More precisely, it is divided into three distinct phases. In the first one, we use a local search algorithm that finds an optimal MN searching the MN space. It is well-known that only chordal MNs are precisely representable by BNs (Pearl, 1988). As it is unlikely that the MN we find at the end of the first phase is actually chordal, its transformation into a BN must come along with the loss of some CIs. Finding a set of CIs that can be dispensed with to map the MN into a BN while fitting data as best as possible is not a trivial task. Phase 2 exploits the relationship between BNs and their moral graphs to transform the MN into a BN whose moral graph is not far from the MN obtained at the end of phase 1. Then, in a third phase, this BN is refined into one that better fits data. This last phase uses GES second step. The whole learning algorithm preserves GES theoretical optimality guarantee. Furthermore, the BN at the end of phase 2, which possesses interesting theoretical properties, is obtained in polynomial time. Phase 3 is exponential but has an anytime property. Preliminary experimental results suggest that our learning algorithm is faster than GES and produces better quality solutions.

The paper is organized as follows. Section 2 provides some background on MNs and BNs. Then, Section 3 describes how we obtain the optimal MN. Section 4 shows how it can be converted into a BN. Finally Section 5 mentions some related work and presents some experimental results.

## 2 Background

Let $\mathcal{V}$ be a set of random variables with joint probability distribution $P(\mathcal{V})$. Variables are denoted by capitalized letters (other than $P$) and sets of variables (except $\mathcal{V}$) by bold letters.

Markov and Bayes nets are both composed of: i) a graphical structure whose nodes are the variables in $\mathcal{V}$ (hereafter, we indifferently refer to nodes and their corresponding variables) and ii) a set of numerical parameters. The structure encodes probabilistic CIs among variables and then defines a family of probability distributions. The set of parameters, whose form depends on the structure, defines a unique distribution among this family and assesses it numerically. Once a structure is learned from data, its parameters are assessed, generally by maximizing data likelihood given the model.

A MN structure is an undirected graph while a BN one is a directed acyclic graph (DAG). MNs graphically encode CIs by separation. In an undirected graph $\mathcal{G}$, two nodes $X$ and $Y$ are said to be *separated* by a disjoint set $\mathbf{Z}$, denoted by $X \perp_{\mathcal{G}} Y \mid \mathbf{Z}$, if each chain between $X$ and $Y$ has a node in $\mathbf{Z}$. BNs criterion, called d-separation, is defined similarly except that it distinguishes *colliders* on a chain, i.e., nodes with their two adjacent arcs on the chain directed toward them. In a DAG, $X$ and $Y$ are said to be *d-separated* by $\mathbf{Z}$, denoted by $X \perp_{\mathcal{B}} Y \mid \mathbf{Z}$, if for each chain between $X$ and $Y$ there exists a node $S$ s.t. if $S$ is a collider on the chain, then neither $S$ nor any of its descendant is in $\mathbf{Z}$, else $S$ is in $\mathbf{Z}$. Extending Pearl's terminology (Pearl, 1988), we will say that a graph $\mathcal{G}$, directed or not, is an *I-map* (*I* standing for *independency*), implicitly relative to $P$, if $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z} \Longrightarrow \mathbf{X} \perp\!\!\!\perp_P \mathbf{Y} \mid \mathbf{Z}$. We will also say that a graph $\mathcal{G}$ is an I-map of another graph $\mathcal{G}'$ when $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z} \Longrightarrow \mathbf{X} \perp_{\mathcal{G}'} \mathbf{Y} \mid \mathbf{Z}$. When two structures are I-maps of one another, they represent the same family and are thus said to be *equivalent*.

The complete graph, which exhibits no separation assertion, is a structure containing all the possible distributions and is, as such, always an I-map. Of course, in a learning prospect, our aim is to recover from data an I-map as sparse as possible. More pre-

cisely, in both MN and BN frameworks, an I-map $\mathcal{G}$ is said to be *optimal* (w.r.t. inclusion) relatively to $P$ if there exists no other I-map $\mathcal{G}'$ such that i) $\mathcal{G}$ is an I-map of $\mathcal{G}'$ and ii) $\mathcal{G}$ is not equivalent to $\mathcal{G}'$.

Though they are strongly related, BNs and MNs are not able to represent precisely the same independence shapes. MNs are mostly used in the image processing and computer vision community (Pérez, 1998), while BNs are particularly used in the AI community working on modeling and prediction tools like expert systems (Cowell et al., 1999). In the latter prospect, BNs are mostly preferred for two reasons. First, the kind of independences they can express using arcs orientations are considered more interesting than the independence shapes only representable by MNs (Pearl, 1988). Secondly, as BNs parameters are probabilities (while those of MNs are non-negative potential functions without any real actual meaning), they are considered easier to assess, to manipulate and to interpret. In BNs, the direction is exploited only through *V-structures*, that is, three-node chains whose central node is a collider the neighbors of which are not connected by an arc. This idea is expressed in a theorem (Pearl, 1988) stating that two BNs are equivalent if and only if they have the same V-structures and the same *skeleton*, where the skeleton of a DAG is the undirected graph resulting from the removal of its arcs direction.

MNs are unable to encode V-structure information since it is a directed notion. As a DAG without V-structure is chordal (i.e. triangulated), it is not surprising that a result (Pearl et al., 1989) states that independences of a family of distributions can be represented by both MNs and BNs frameworks if and only if the associated structure is chordal. In the general case where the graph is not chordal, it is possible to get an optimal MN from a BN by *moralization*. The *moral graph* of a DAG is the undirected graph obtained by first adding edges between non adjacent nodes with a common child (i.e. the extremities of V-structures) and then removing the arcs orientations. It is easily seen that the moral graph of a BN is an optimal undirected I-map of this graph, even if some independences of the BN have been necessarily loosed in the transformation. The converse, i.e. getting a directed I-map from a MN, is less easy and will be addressed in Section 4.

Like most works aiming to recover an optimal structure from data, we will assume that the underlying distribution $P$ is *DAG-isomorph*, i.e. that there exists a DAG $\mathcal{B}^*$ encoding exactly the independences of $P$ (s.t. $\mathbf{X} \perp_{\mathcal{B}^*} \mathbf{Y} \mid \mathbf{Z} \iff \mathbf{X} \perp\!\!\!\perp_P \mathbf{Y} \mid \mathbf{Z}$). Under this assumption, $\mathcal{B}^*$ and equivalent BNs are obviously optimal. Using the axiomatic in (Pearl, 1988), it can be shown that, in the MN framework, the DAG-isomorphism hypothesis entails the so-called intersection property, leading to the existence of a unique optimal MN, say $\mathcal{G}^*$ (Pearl, 1988). Moreover, $\mathcal{G}^*$ is the moral graph of $\mathcal{B}^*$.

## 3 Markov network search

Searching the BN or the EC space is often performed as an optimization process, that is, the algorithm looks for a structure optimizing some goodness of fit measure, the latter being a decomposable scoring function that involves maximum likelihood estimates. For MNs, the computation of these estimates is hard and requires time-expensive methods (Murray and Ghahramani, 2004) (unless the MN is triangulated, which is not frequent). Hence score-based exploration strategies seem inappropriate for MN searches. Using statistical CI tests and combining them to reveal the MN's edges seems a more promising approach. This is the one we follow here.

***Algorithm*** LEARNMN
**Input :** database
**Output :** moral graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_2)$

1. $\mathcal{E}_1 \leftarrow \emptyset$
2. **foreach** edge $(X, Y) \notin \mathcal{E}_1$ **do**
   search, if necessary, a new $\mathbf{S_{XY}}$
   s.t. $X \perp_{(\mathcal{V},\mathcal{E}_1)} Y \mid \mathbf{S_{XY}}$
3. **if** $\exists\, (X, Y) \notin \mathcal{E}_1$ s.t. $X \not\perp\!\!\!\perp Y | \mathbf{S_{XY}}$ **then**
4. add edge $(X, Y)$ to $\mathcal{E}_1$ and go to line 2
5. $\mathcal{E}_2 \leftarrow \mathcal{E}_1$
6. **foreach** edge $(X, Y) \in \mathcal{E}_2$ **do**
   search, if necessary, a new $\mathbf{S_{XY}}$
   s.t. $X \perp_{(\mathcal{V},\mathcal{E}_2\setminus\{(X,Y)\})} Y \mid \mathbf{S_{XY}}$
7. **if** $\exists\, (X, Y) \in \mathcal{E}_2$ s.t. $X \perp\!\!\!\perp Y | \mathbf{S_{XY}}$ **then**
8. remove edge $(X, Y)$ from $\mathcal{E}_2$ and go to line 6
9. **return** $\mathcal{G} = (\mathcal{V}, \mathcal{E}_2)$

***End of algorithm***

Unlike most works where learning a MN amounts to independently learn the Markov blanket of all the variables, we construct a MN in a local search manner. Algorithm LEARNMN consists in two consecutive phases: the first one adds edges to the empty graph until it converges toward a graph $\mathcal{G}_1 =$

$(\mathcal{V}, \mathcal{E}_1)$. Then it removes from $\mathcal{G}_1$ as many edges as possible, hence resulting in a graph $\mathcal{G}_2 = (\mathcal{V}, \mathcal{E}_2)$.

At each step of phase 1, we compute the dependence of each pair of non-adjacent nodes $(X, Y)$ conditionally to any set $\mathbf{S_{XY}}$ separating them in the current graph. We determine the pair with the strongest dependence (using a normalized difference to the $\chi^2$ critical value) and we add the corresponding edge to the graph and update separators.

**Lemma 1.** *Assuming DAG-isomorphism and sound statistical tests, graph $\mathcal{G}_1$ resulting for the first phase of* LEARNMN *is an I-map.*

*Sketch of proof.* At the end of phase 1, $\forall\ (X, Y) \notin \mathcal{E}_1$, there exists $\mathbf{S_{XY}}$ s.t. $X \perp_{\mathcal{G}_1} Y \mid \mathbf{S_{XY}}$ and $X \perp\!\!\!\perp Y \mid \mathbf{S_{XY}}$. By DAG-isomorphism, it can be shown that $\mathcal{G}_1$ contains the skeleton of $\mathcal{B}^*$, and, then, that it also contains its moralization edges. □

In the second phase, which takes an I-map $\mathcal{G}_1$ as input, LEARNMN detects the superfluous edges in $\mathcal{G}_1$ and iteratively removes them.

**Proposition 1.** *Assuming DAG-isomorphism and sound statistical tests, the graph $\mathcal{G}_2$ returned by* LEARNMN *is the optimal MN $\mathcal{G}^*$.*

*Sketch of proof.* At the end of phase 2, $\forall\ (X, Y) \in \mathcal{G}_2$, there exists $\mathbf{S_{XY}}$ s.t. $X \perp_{(\mathcal{V}, \mathcal{E}_2 \setminus \{(X,Y)\})} Y \mid \mathbf{S_{XY}}$ and $X \not\perp\!\!\!\perp Y \mid \mathbf{S_{XY}}$. We show using DAG-isomorphism that this phase cannot delete any edge in $\mathcal{G}^*$ and, then, that it discards all other edges. □

It is well known that the main shortcoming of independence test-based methods is the unreliability of high-order tests, so we must keep sets $\mathbf{S_{XY}}$ as small as possible. Finding small $\mathbf{S_{XY}}$ requires a more sophisticated approach than simply using the set of neighbors of one of $X$ or $Y$. Actually, the best set is the smallest one that cuts all the paths between $X$ and $Y$. This can be computed using a min-cut in a max-flow problem (Tian et al., 1998). Although the set computed is then optimal w.r.t. the quality of the independence test, it has a major drawback: computing a min-cut for all possible pairs of nodes $(X, Y)$ is too prohibitive when the graph involves numerous variables. Moreover, pairs of close nodes require redundant computations. An attractive alternative results from the similarities between min-cuts and junction trees: a min-cut separates the MN into several connected components, just as a separator cuts a junction tree into distinct connected components. Hence, we propose a method, based on join trees (Jensen, 1996), which computes the separators for all pairs at the same time in $O(|\mathcal{V}|^4)$, as compared to $O(|\mathcal{V}|^5)$ in (Tian et al., 1998). Although the separators we compute are not guaranteed to be optimal, in practice they are most often small. Here is the key idea: assume $\mathcal{G}_0$ is triangulated and a corresponding join tree $\mathcal{J}_0$ is computed. Then two cases can obtain: i) $X$ and $Y$ belong to the same clique, or ii) they belong to different cliques of $\mathcal{J}_0$. In the second case, let $\mathbf{C_0}, \ldots, \mathbf{C_k}$ be the smallest connected set of cliques such that $X \in \mathbf{C_0}$ and $Y \in \mathbf{C_k}$, i.e., $\mathbf{C_0}$ and $\mathbf{C_k}$ are the nearest cliques containing $X$ and $Y$. Then any separator on the path $\mathbf{C_0}, \ldots, \mathbf{C_k}$ cuts all the paths between $X$ and $Y$ in $\mathcal{G}_0$ and, thus, can act as an admissible set $\mathbf{S_{XY}}$. Assuming the triangulation algorithm produces separators as small as possible, selecting the smallest one should keep sets $\mathbf{S_{XY}}$ sufficiently small to allow independence tests. As for the first case, there is no separator between $X$ and $Y$, so the junction tree is helpless. However, we do not need assigning to $\mathbf{S_{XY}}$ all the neighbors of $X$ or $Y$, but only those that belong to the same biconnected component as $X$ and $Y$ (as only those can be on the chains between $X$ and $Y$). Such components can be computed quickly by depth first search algorithms. However, as we shall see, the triangulation algorithm we used determines them implicitly.

In order to produce triangulations in polynomial time (determining an optimal one is NP-hard), we used the simplicial and almost-simplicial rules advocated by (Eijkhof et al., 2002) as well as some heuristics. These rules give high-quality triangulations but are time-consuming. So, to speed-up the process, we used incremental triangulations as suggested by (Flores et al., 2003). The idea is to update the triangulation only in the maximal prime subgraphs of $\mathcal{G}_0$ involved in the modifications resulting from LEARNMN's lines 4 and 8. In addition to the join tree, a join tree of max prime subgraphs is thus maintained by the algorithm. It turns out that merging in this tree the adjacent cliques that are linked by a separator containing more than one node of $\mathcal{V}$ precisely produces $\mathcal{G}_0$'s biconnected components.

Once the join tree constructed, extracting for each pair of nodes $(X, Y)$ not belonging to the same clique its minimal separator is achieved by the col-

lect/distribute algorithms below. In the latter, messages $\mathbf{M_{ij}}$ transmitted from a clique $\mathbf{C_j}$ to $\mathbf{C_i}$ are vectors of pairs $(X, \mathbf{S})$ such that the best conditioning sets between nodes $Y$ in $C_i \backslash (C_i \cap C_j)$ and $X$ is $\mathbf{S}$. An illustrative example is given on Figure 1: messages near solid arrows result from COLLECT($\{BCD\}$,$\{BCD\}$) and those in dashed arrows from DISTRIBUTE($\{BCD\}$,$\{BCD\}$,$\emptyset$).

***Algorithm*** COLLECT
**Input :** pair $(\mathbf{C_i}, \mathbf{C_j})$
**Output :** message $\mathbf{M_{ij}}$

1. $\mathbf{M_{ij}} \leftarrow$ an empty vector message
2. **foreach** neighbor $\mathbf{C_k}$ of $\mathbf{C_i}$ except $\mathbf{C_j}$ **do**
3. $\quad \mathbf{M_{ik}} \leftarrow$ Collect($\mathbf{C_k}, \mathbf{C_i}$)
4. $\quad$ **foreach** pair $(X, \mathbf{S})$ in $\mathbf{M_{ik}}$ **do**
5. $\quad\quad$ **foreach** node $Y$ in $\mathbf{C_i} \backslash (\mathbf{C_i} \cap \mathbf{C_k})$ **do**
6. $\quad\quad\quad$ **if** $\mathbf{S_{XY}} \neq \mathbf{S}$ **then**
7. $\quad\quad\quad\quad \mathbf{S_{XY}} \leftarrow \mathbf{S}$
8. $\quad\quad$ **done**
9. $\quad\quad$ add $(X, min(\mathbf{C_i} \cap \mathbf{C_k}, \mathbf{S}))$ to $\mathbf{M_{ij}}$
10. $\quad$ **done**
11. **done**
12. **foreach** node $X$ in $\mathbf{C_i} \backslash (\mathbf{C_i} \cap \mathbf{C_j})$ **do**
13. $\quad$ add $(X, \mathbf{C_i} \cap \mathbf{C_j})$ to $\mathbf{M_{ij}}$
14. **return** $\mathbf{M_{ij}}$

***End of algorithm***

***Algorithm*** DISTRIBUTE
**Input :** pair $(\mathbf{C_i}, \mathbf{C_j})$, message $\mathbf{N_{ij}}$
**Output :**

1. **foreach** pair $(X, \mathbf{S})$ in $\mathbf{N_{ij}}$ **do**
2. $\quad$ **foreach** node $Y$ in $\mathbf{C_i} \backslash (\mathbf{C_i} \cap \mathbf{C_j})$ **do**
3. $\quad\quad$ **if** $\mathbf{S_{XY}} \neq \mathbf{S}$ **then**
4. $\quad\quad\quad \mathbf{S_{XY}} \leftarrow \mathbf{S}$
5. $\quad$ **done**
6. **done**
7. **foreach** neighbor $\mathbf{C_k}$ of $\mathbf{C_i}$ except $\mathbf{C_j}$ **do**
8. $\quad \mathbf{N} \leftarrow$ empty message vector
9. $\quad$ **foreach** pair $(X, \mathbf{S})$ in $\mathbf{N_{ij}}$ **do**
10. $\quad\quad$ add $(X, min(\mathbf{C_i} \cap \mathbf{C_k}, \mathbf{S}))$ to $\mathbf{N}$
11. $\quad$ call Distribute $(\mathbf{C_k}, \mathbf{C_i}, \mathbf{N})$
12. $\quad \mathbf{N'} \leftarrow$ message $\mathbf{M_{ik}}$ sent during collect
13. $\quad$ **foreach** pair $(X, \mathbf{S})$ in $\mathbf{N'}$ **do**
14. $\quad\quad$ add $(X, min(\mathbf{C_i} \cap \mathbf{C_k}, \mathbf{S}))$ to $\mathbf{N_{ij}}$
15. **done**

***End of algorithm***

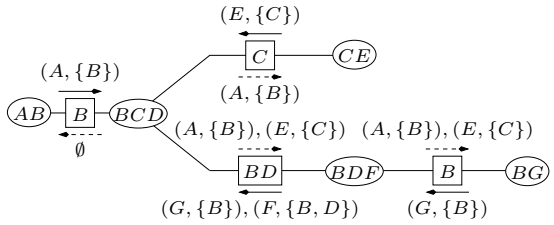



Figure 1: Messages sent during Collect/diffusion.

Because we use a polynomial algorithm to triangulate the maximal prime subgraphs, the complexity of the whole process recovering the optimal MN is also polynomial. Hence, under DAG-isomorphism, we obtain the moral graph of the optimal BN in polynomial time. As the first step of inference algorithms consists in moralizing the BN, and triangulating this moral graph to produce a secondary structure well-suited for efficient computations, the MN we obtain is optimal w.r.t. inference.

## 4 From the Markov Net to the Bayes Net

Once the MN is obtained, we transform it into a BN $\mathcal{B}_0$ using algorithm MN2BN.

***Algorithm*** MN2BN
**Input :** UG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
**Output :** DAG $\mathcal{B} = (\mathcal{V}', \mathcal{A})$

1. $\mathcal{A} \leftarrow \emptyset$ ; $\mathcal{V}' \leftarrow \mathcal{V}$
2. **While** $|\mathcal{V}| > 1$ **Do**
3. $\quad$ Choose $X \in \mathcal{V}$ in a single clique of $\mathcal{G}$
4. $\quad \mathbf{E} \leftarrow \{Y \in \mathcal{V} : (Y, X) \in \mathcal{E}\}$
5. $\quad$ **For all** $Y \in \mathbf{E}$ **Do**
6. $\quad\quad \mathcal{E} \leftarrow \mathcal{E} \backslash \{(Y, X)\}$ and $\mathcal{A} \leftarrow \mathcal{A} \cup \{(Y \rightarrow X)\}$
7. $\quad \mathcal{V} \leftarrow \mathcal{V} \backslash \{X\}$
8. $\quad$ **For all** $(Y, Z) \in \mathbf{E} \times \mathbf{E}$ **Do**
9. $\quad\quad \mathcal{G} \leftarrow$ DEMORALIZE$(\mathcal{G}, (Y, Z))$
10. **Done**
11. **Return** $\mathcal{B} = (\mathcal{V}', \mathcal{A})$

***End of algorithm***

***Algorithm*** DEMORALIZE
**Input :** UG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ; Edge $(X, Y) \in \mathcal{E}$
**Output :** UG $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$

1. $\mathcal{E}' \leftarrow \mathcal{E} \backslash \{(X, Y)\}$
2. Search $\mathbf{Z} \subset \mathcal{V}$ s.t. $X \perp_{(\mathcal{V}, \mathcal{E}')} Y \mid \mathbf{Z}$
3. **If** $X \perp\!\!\!\perp Y | \mathbf{Z}$ **Then Return** $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$
4. **Else Return** $\mathcal{G}' = (\mathcal{V}, \mathcal{E})$

***End of algorithm***

Before stating the properties of MN2BN, let us illustrate it with an example. Consider the graphs in Figure 2. Assuming a DAG-isomorph distribution, $\mathcal{B}^*$ represents the optimal BN. Suppose we obtained the optimal MN $\mathcal{G}^*$: we now apply MN2BN to it. In $\mathcal{G} = \mathcal{G}^*$, $F$ and $R$ belong to a single clique. Assume MN2BN chooses $F$. After saving its neighbors set in $\mathcal{G}$ (l.4), $F$ is removed from $\mathcal{G}$ (l.7) as well as its adjacent edges (l.6), which results in a new current graph $\mathcal{G} = \mathcal{G}_1$. The empty DAG $\mathcal{B}$ is altered by adding arcs from these neighbors toward $F$, hence resulting in the next current DAG $\mathcal{B} = \mathcal{B}_1$. By directing $F$'s adjacent arcs toward $F$, we may create

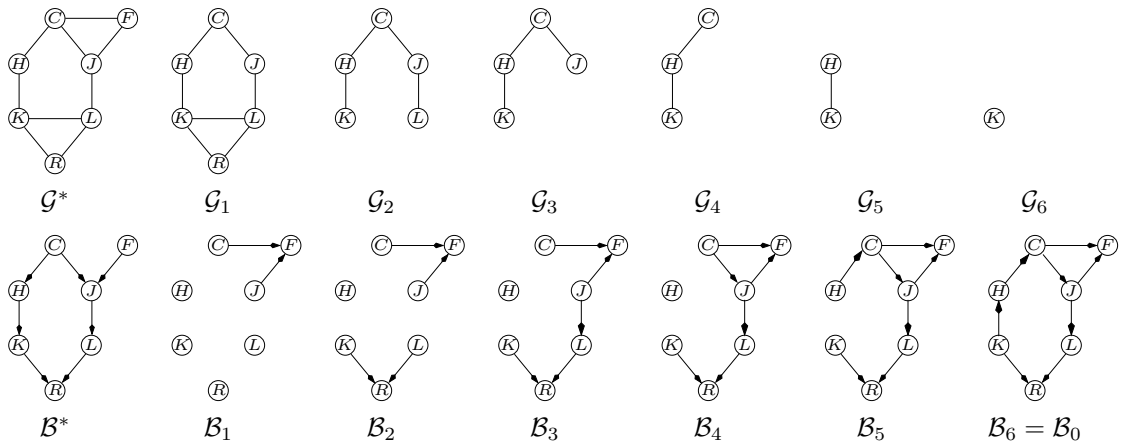



Figure 2: A MN2BN run example

V-structures, hence some of the edges remaining in $\mathcal{G}$ between $F$'s neighbors may correspond to moralization edges and may thus be safely discarded. To this end, for each candidate edge, DEMORALIZE tests whether its deletion would introduce spurious independence in the DAG by searching a separator in the current remaining undirected graph $\mathcal{G}$. If not, we discard it from $\mathcal{G}$. In our example, there is only one candidate edge, $(C, J)$. We therefore compute a set separating $C$ from $J$ in $\mathcal{G}_1$ without $(C, J)$, say $K$, and test if $C \perp\!\!\!\perp J \mid K$. As this independence does not hold in a distribution exactly encoded by $\mathcal{B}^*$, the test fails and the edge is kept. For the second iteration, only node $R$ belongs to a single clique. The process is similar but, this time, moralization edge $(K, L)$ can be discarded. The algorithm then goes on similarly and finally returns $\mathcal{B}_6 = \mathcal{B}_0$. It is easily seen that $\mathcal{B}_0$ is a DAG which is an I-map of $\mathcal{B}^*$ and that its moral graph is $\mathcal{G}^*$. Note that by mapping $\mathcal{G}^*$ into $\mathcal{B}_6$, not all moralization edges have been identified. For instance, $(C, F)$ was not.

Now, let us explain why MN2BN produces DAGs closely related to the optimal BN we look for.

**Proposition 2.** *Assuming a DAG-isomorph distribution and sound statistical tests, if* MN2BN *is applied to $\mathcal{G}^*$, the returned graph $\mathcal{B}_0$ is a DAG that is an I-map of $\mathcal{B}^*$, and its moral graph is $\mathcal{G}^*$. Moreover, the algorithm runs in polynomial time.*

*Sketch of proof.* By induction on $\mathcal{V}$, we prove that, at each iteration, $\mathcal{G}$ is the moral graph of a subgraph of $\mathcal{B}^*$. This entails the existence of a node in a single clique at each iteration. Then we prove that given a DAG $\mathcal{B}$, its moral graph $\mathcal{G}$ and a node $X$ belonging to a single clique of $\mathcal{G}$, there exists a DAG $\mathcal{B}'$ s.t.: i) $\mathcal{B}'$ is an I-map of $\mathcal{B}$, ii) $\mathcal{G}$ is the moral graph of $\mathcal{B}'$ and iii) $X$ has no child in $\mathcal{B}'$. Finally, we prove the proposition by induction on $\mathcal{V}$. □

$\mathcal{B}_0$ encodes at least as many independences as $\mathcal{G}^*$ and possibly more if some moralization edges have been discarded. In the case where MN2BN selects nodes in the inverse topological order of $\mathcal{B}^*$, $\mathcal{B}_0$ is actually $\mathcal{B}^*$. However, this should not happen very often and recovering $\mathcal{B}^*$ requires in general to refine $\mathcal{B}_0$ by identifying all remaining hidden V-structures. Under DAG-isomorph distribution and sound statistical tests, the second phase of GES (Chickering, 2002) is known to transform an I-map of the generative distribution into $\mathcal{B}^*$. Applied to $\mathcal{B}_0$, it can therefore be used as the refinement phase of our algorithm. This one has an exponential complexity but benefits from an anytime property, in the sense that it proceeds by constructing a sequence of BNs sharing the same moral graph and the quality of which converges monotonically from $\mathcal{B}_0$ to $\mathcal{B}^*$. This last phase preserves the GES score-based optimality.

With real-world databases, LEARNMN can fail to recover precisely $\mathcal{G}^*$. In this case, departing from our theoretical framework, we lose our guarantee of accuracy. We also lose the guarantee to always find a node belonging to a single clique. However, MN2BN can easily be modified to handle this situation: if no node is found on line 3, just add edges to $\mathcal{G}$ so that a given node forms a clique with its neighbors. Whatever the node chosen, $\mathcal{B}_0$ is guaranteed

to be an I-map of the distribution represented by the input MN. However, the node should be carefully chosen to avoid deviating too much from the MN passed in argument to MN2BN, as each edge addition hides conditional independences.

## 5 Related works and experiments

In this paper, we have presented an algorithm that exploits attractive features of the MN space to quickly compute an undirected I-map close to the optimal one. This graph is then directed and fed to GES second phase to obtain an optimal BN. The key idea is to learn as much as possible the polynomially accessible information of the generative distribution, namely its undirected part, and postpone to the end the task concentrating the learning computational complexity, namely the V-structure recovery. Most algorithms of the constraint-based approach, like IC (Verma and Pearl, 1990) or PC (Spirtes et al., 2000), deal at some stage with undirected learning but they do not explicitly separate this stage from the directed one. That is, they generally search a power set to extract V-structures information before achieving a whole undirected search. In (Dijk et al., 2003), the two phases are disjoint but the authors are more concerned with finding the skeleton (which is not an I-map) rather than the moral graph. As they only allow CI tests conditioned by a set of size no greater than 1, the search is mostly delegated to a directed score-based search. (Cheng et al., 2002) makes much stronger distribution assumptions.

Like GES, under DAG-isomorphism and sound CI test hypotheses, our method is able to find the optimal BN it looks for. However, these hypotheses probably do not hold in practice. To assess the discrepancy between theory and real world, we performed a series of experiments on classical benchmarks of the Bayes Net Repository[1]. For each BN, we generated by logic sampling 10 databases of size 500, 2000 and 20000. For each database, we ran LEARNMN, GES (the WinMine toolkit software) and a K2 implemented with a BIC score and fed with the original BN topological ordering. K2 is thus already aware of some crucial learning information that LMN and GES must discover. For GES, we recorded as time response only the time required for GES phase 1 but, as the toolkit does not provide the network found at the end of phase 1, we used the better one resulting from phase 2. For both GES and K2, we did not consider directly the output BN but its moral graph, to compare it with the MN output by LMN. Note that the GES toolkit was unable to handle some of the graphs. Both time and quality results reported are means and standard deviations for the 10 databases. The first table shows running times in seconds for LMN and GES (on a 3.06GHz PC). The second one shows for each algorithm the number of edges spuriously added (+) and missed (-) w.r.t. the original BN moral graph (whose number of edges appears in the first row).

According to these preliminary tests, we consider our approach as promising. W.r.t GES, we find that LMN is not time-consuming, as GES running time grows much faster than that of LMN when database size increases. As for networks quality, it is noticeable that, for all algorithms, the smaller the database, the weaker the dependences represented in the database and hence the fewer the edges recovered. LMN finds clearly more edges than GES and K2, despite the latter's ordering knowledge which gives it a strong advantage. LMN adds fewer spurious edges than GES but more than K2 (which exploits its ordering knowledge). We think that LMN did not detect some of the unnecessary edges because nodes involved had so many neighbors that $\chi^2$ tests were meaningless, despite our efforts. This suggests alternating multiple phases of edge additions and deletions, so as to avoid situations deteriorating due to big clusters of neighbors. We also should try to tune the $\chi^2$ confidence probability.

[1] `http://compbio.cs.huji.ac.il/Repository`

## References


D. Allen and A. Darwiche. 2003. Optimal time-space tradeoff in probabilistic inference. In *proceedings of IJCAI*, pages 969–975.

J. Cheng, R. Greiner, J. Kelly, D.A. Bell, and W. Liu. 2002. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1–2):43–90.

D.M. Chickering. 2002. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J.

| | | alarm | | barley | | carpo | | hailfinder | | insurance | | mildew | | water | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| size | | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev |
| 20000 | LMN | 3.65 | 0.82 | 9.38 | 0.73 | 11.96 | 2.67 | 15.56 | 1.31 | 3.59 | 0.1 | 3.29 | 0.37 | 7.48 | 0.78 |
| | GES | 22.36 | 1.72 | | | 75.91 | 4.19 | | | 16.74 | 4.22 | | | 18.36 | 3.91 |
| 2000 | LMN | 0.98 | 0.28 | 1.46 | 0.46 | 1.79 | 0.68 | 2.36 | 0.65 | 1.05 | 0.12 | 0.56 | 0.08 | 1.37 | 0.21 |
| | GES | 1.36 | 0 | | | 7.27 | 0.52 | | | 1.5 | 0 | | | 1.55 | 0 |
| 500 | LMN | 0.47 | 0.15 | 0.31 | 0.01 | 0.71 | 0.27 | 1.24 | 0.39 | 0.63 | 0.2 | 0.28 | 0.02 | 0.44 | 0.12 |
| | GES | 0.64 | 0 | | | 0.64 | 0 | | | 0.5 | 0 | | | 0.45 | 0 |

| | | alarm (65) | | barley (126) | | carpo (102) | | hailfinder (99) | | insurance (70) | | mildew (80) | | water (123) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| size | | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev | Avg. | Stdev |
| 20000 | GES + | 13.6 | 2.69 | | | 26.6 | 3.85 | | | 17.6 | 5.14 | | | 17.9 | 2.34 |
| | GES - | 39.7 | 1.1 | | | 38.7 | 4.22 | | | 44.4 | 3.17 | | | 94.6 | 2.15 |
| | LMN + | 2.1 | 1.3 | 15.7 | 1.1 | 9.4 | 1.28 | 9.7 | 3 | 0.4 | 0.49 | 3.2 | 0.75 | 0 | 0 |
| | LMN - | 9.6 | 1.11 | 60.1 | 2.02 | 23.2 | 1.89 | 10.1 | 0.94 | 15.5 | 1.63 | 22.3 | 1.55 | 52.3 | 1.85 |
| | K2 + | 3.2 | 0.75 | 3 | 0 | 3.1 | 1.58 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | K2 - | 19.7 | 1.79 | 87.8 | 0.6 | 31.1 | 2.43 | 26 | 0 | 19.3 | 1.1 | 59 | 0 | 71.6 | 2.58 |
| 2000 | GES + | 14 | 0.63 | | | 11.8 | 3.6 | | | 11.8 | 2.32 | | | 3.9 | 1.14 |
| | GES - | 62 | 0.77 | | | 66.4 | 2.2 | | | 59.2 | 1.47 | | | 110.3 | 1.73 |
| | LMN + | 3.4 | 0.8 | 13.3 | 2.19 | 7.5 | 3.35 | 2.2 | 1.17 | 0.3 | 0.46 | 4.5 | 1.2 | 0.2 | 0.4 |
| | LMN - | 21.3 | 2.33 | 79 | 1.73 | 33.3 | 2.61 | 15.4 | 1.28 | 24.9 | 2.12 | 40.4 | 1.2 | 75 | 3.87 |
| | K2 + | 3.2 | 0.6 | 3.1 | 0.3 | 2.3 | 0.78 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | K2 - | 34.9 | 1.37 | 98.3 | 0.78 | 45.7 | 3.72 | 50.2 | 1.83 | 38.2 | 1.47 | 64.4 | 1.28 | 98.2 | 2.48 |
| 500 | GES + | 3.9 | 1.64 | | | 10.3 | 2.28 | | | 5.3 | 1 | | | 0.5 | 0.5 |
| | GES - | 65 | 0 | | | 89.1 | 2.07 | | | 65.1 | 1.14 | | | 118.2 | 1.25 |
| | LMN + | 7.1 | 2.26 | 8.2 | 1.25 | 5.9 | 2.12 | 2 | 1.26 | 1.1 | 0.94 | 4.4 | 1.5 | 0.4 | 0.92 |
| | LMN - | 31.5 | 2.01 | 90.2 | 1.08 | 54.2 | 2.52 | 33.6 | 3.07 | 34.7 | 1.68 | 56.8 | 0.98 | 88.3 | 2.24 |
| | K2 + | 3.7 | 0.78 | 3.4 | 0.49 | 4.5 | 2.29 | 0.1 | 0.3 | 0.3 | 0.46 | 0 | 0 | 0 | 0 |
| | K2 - | 42.7 | 1 | 105.2 | 0.98 | 60.8 | 2.44 | 65.5 | 1.2 | 47.2 | 0.98 | 74.8 | 0.75 | 101.6 | 1.2 |


Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer-Verlag.

S. van Dijk, L. van der Gaag, and D. Thierens. 2003. A skeleton-based approach to learning Bayesian networks from data. In *Proceedings of PKDD*.

F. van den Eijkhof, H.L. Bodlaender, and A.M.C.A. Koster. 2002. Safe reduction rules for weighted treewidth. Technical Report UU-CS-2002-051, Utrecht University.

J. Flores, J. Gamez, and K. Olesen. 2003. Incremental compilation of bayesian networks. In *Proceedings of UAI*, pages 233–240.

S.B. Gillispie and M.D. Perlman. 2001. Enumerating Markov equivalence classes of acyclic digraphs models. In *Proceedings of UAI*, pages 171–177.

D. Heckerman, D. Geiger, and D.M. Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243.

F.V. Jensen. 1996. *An Introduction to Bayesian Networks*. Springer-Verlag, North America.

W. Lam and F. Bacchus. 1993. Using causal information and local measures to learn Bayesian networks. In *Proceedings of UAI*, pages 243–250.

A.L. Madsen and F.V. Jensen. 1999. Lazy propagation: A junction tree inference algorithm based on lazy inference. *Artificial Intelligence*, 113:203–245.

P. Munteanu and M. Bendou. 2001. The EQ framework for learning equivalence classes of Bayesian networks. In *Proceedings of the IEEE International Conference on Data Mining*, pages 417–424.

I. Murray and Z. Ghahramani. 2004. Bayesian learning in undirected graphical models: Approximate MCMC algorithms. In *Proceedings of UAI*.

J. Pearl, D. Geiger, and T.S. Verma. 1989. The logic of influence diagrams. In R.M. Oliver and J.Q. Smith, editors, *Influence Diagrams, Belief Networks and Decision Analysis*. John Wiley and Sons.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman.

P. Pérez. 1998. Markov random fields and images. *CWI Quarterly*, 11(4):413–437.

R.W. Robinson. 1973. Counting labeled acyclic digraphs. In F. Harary, editor, *New directions in Graph Theory*, pages 239–273. Academic Press.

P. Spirtes, C. Glymour, and R. Scheines. 2000. *Causation, Prediction, and Search*. Springer Verlag.

J. Tian, A. Paz, and J. Pearl. 1998. Finding minimal d-separators. Technical Report TR-254, Cognitive Systems Laboratory.

T.S. Verma and J. Pearl. 1990. Causal networks : Semantics and expressiveness. In *Proceedings of UAI*, pages 69–76.

# Estimation of linear, non-gaussian causal models in the presence of confounding latent variables

Patrik O. Hoyer[1], Shohei Shimizu[2,1] and Antti J. Kerminen[1]

[1)] HIIT Basic Research Unit
Department of Computer Science
University of Helsinki
Finland

[2)] Learning and Inference Group
The Institute of Statistical Mathematics
Japan

**Abstract**

The estimation of linear causal models (also known as structural equation models) from data is a well-known problem which has received much attention in the past. Most previous work has, however, made an explicit or implicit assumption of gaussianity, limiting the identifiability of the models. We have recently shown (Shimizu et al., 2005; Hoyer et al., 2006) that for non-gaussian distributions the full causal model can be estimated in the no hidden variables case. In this contribution, we discuss the estimation of the model when confounding latent variables are present. Although in this case uniqueness is no longer guaranteed, there is at most a finite set of models which can fit the data. We develop an algorithm for estimating this set, and describe numerical simulations which confirm the theoretical arguments and demonstrate the practical viability of the approach. Full Matlab code is provided for all simulations.

## 1 Introduction

The ultimate goal of much of the empirical sciences is the discovery of *causal relations*, as opposed to just correlations, between variables. That is, one is not only interested in making predictions based on observations; one also wants to predict what will happen if one intervenes and changes something in the system.

In many cases, causal effects can be estimated using controlled randomized experiments. Unfortunately, however, in many fields of science and in many studies it is not always possible to perform controlled experiments. Often it can be very costly, unethical, or even technically impossible to directly control the variables whose causal effects we wish to learn. In such cases one must rely on observational studies combined with prior information and reasonable assumptions to learn causal relationships. This has been called the *causal discovery* problem (Pearl, 2000; Spirtes et al., 2000).

Linear causal models, also known as structural equation models, can be thought of as the simplest possible causal models for continuous-valued data, and they have been the target of much research over the past decades, see e.g. (Bollen, 1989; Silva et al., 2006) and references therein. The bulk of this research has, however, made an implicit assumption of gaussian data (i.e. normally distributed data). Although the methods that have been developed work for any distributions, they have been limited in their estimation abilities by what can be accomplished for gaussian data. Fortunately, in the real world, many variables are inherently non-gaussian, and in such a case one can obtain much stronger results than for the gaussian case. In earlier work (Shimizu et al., 2005; Hoyer et al., 2006) we showed that if the variables involved are non-normally distributed, yet linearly related to each other, the complete causal graph is identifiable from non-experimental data if no confounding hidden variables are present. In the present paper we show what can be done if this last assumption

is violated. Although in this case uniqueness is no longer guaranteed, there is at most a finite set of models which can fit the data.

This paper is organized as follows. First, in sections 2 and 3, we define the linear causal models that are the focus of this study. Section 4 describes how to estimate the causal model from data. Section 5 provides some simulations confirming the practical viability of the approach. Finally, sections 6 and 7 discuss future work and state some conclusions.

## 2 Linear causal models

Assume that we observe data generated by a linear, non-gaussian, acyclic model (i.e. a LiNGAM-process (Shimizu et al., 2005; Hoyer et al., 2006)) but that we only observe a subset of the involved variables. That is, the process has the following properties:

1. The full set of variables (including unobserved variables) $x_i,\ i = \{1 \dots m\}$ can be arranged in a *causal order*, such that no later variable causes any earlier variable. We denote such a causal order by $k(i)$. That is, the generating process is *recursive* (Bollen, 1989), meaning it can be represented by a *directed acyclic graph* (DAG) (Pearl, 2000; Spirtes et al., 2000).

2. The value assigned to each variable $x_i$ is a *linear function* of the values already assigned to the earlier variables, plus a 'disturbance' (error) term $e_i$, and plus an optional constant term $c_i$, that is

$$x_i = \sum_{k(j)<k(i)} \tilde{b}_{ij} x_j + e_i + c_i. \qquad (1)$$

3. The disturbances $e_i$ are all zero-mean continuous random variables with *non-gaussian* distributions of non-zero variances, and the $e_i$ are independent of each other, i.e. $p(e_1, \dots, e_m) = \prod_i p_i(e_i)$.

4. The *observed variables* is a subset of the $x_i$. We denote the set containing the indices of the observed variables by $J \subseteq \{1, \dots, m\}$. In other words, our data set contains only the $x_j,\ j \in J$.

Figure 1a shows an example of such a *latent variable LiNGAM model*.

An additional assumption not needed in our earlier work (Shimizu et al., 2005; Hoyer et al., 2006) but useful for the purposes of this paper is the requirement that the generating network is *stable* (Pearl, 2000) such that there is no exact canceling of effects. That is, if there is a directed path from $x_i$ to $x_j$ then $x_i$ has a causal effect on $x_j$. This condition has also been termed *faithfulness* (Spirtes et al., 2000), and in our context implies that when multiple causal paths exist from one variable to another their combined effect does not equal exactly zero.

Finally, we assume that we are able to observe a large number of data vectors $\mathbf{x}$ (which contain the observed variables $x_j,\ j \in J$), and that each is generated according to the above described process, with the same set of observed variables $J$, same causal order $k(i)$, same coefficients $\tilde{b}_{ij}$, same constants $c_i$, and the disturbances $e_i$ sampled independently from the same distributions.

The key difference to our previous work (Shimizu et al., 2005; Hoyer et al., 2006) is that we here allow *unobserved confounding variables*: hidden variables which affect multiple observed variables and hence are potentially problematic for any causal analysis. The key difference to other existing research involving linear models with latent variables, such as (Bollen, 1989; Silva et al., 2006), is our assumption of *non-gaussian* variables, allowing us to utilize methods based on higher-order statistics.

## 3 Canonical models

Consider the example model shown in Figure 1a. It should be immediately clear that it is impossible to estimate the full generating model from a sample of data vectors $\mathbf{x} = (x_1, x_2, x_3, x_5, x_8)^T$. This can be seen most clearly by the fact that the data generating model contains a hidden variable ($x_7$) with no observed descendants; detecting the presence of such a variable from our data is obviously not feasible since it has absolutely no effect on the observed data. Fortunately, the impossibility of detecting $x_7$ and of estimating the strengths of

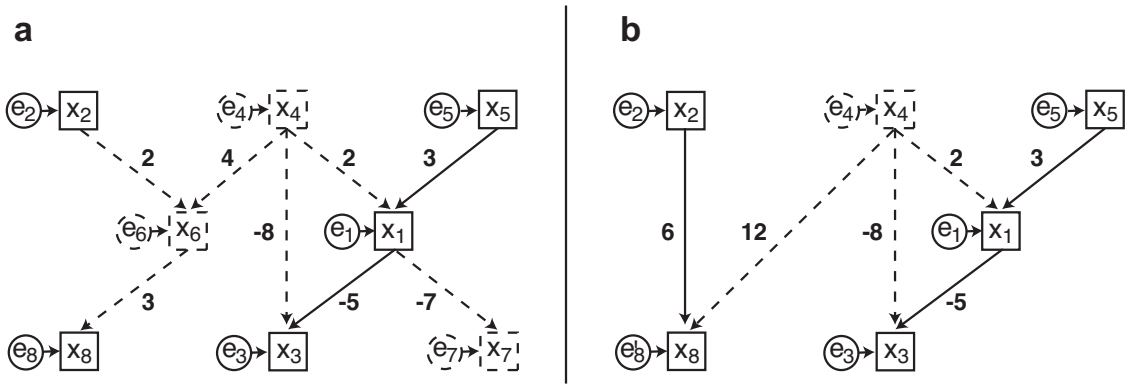



Figure 1: **(a)** An example of a latent variable LiNGAM model. The diagram corresponds to the following data generation process: $x_2 := e_2$, $x_4 := e_4$, $x_5 := e_5$, $x_6 := 2x_2 + 4x_4 + e_6$, $x_1 := 2x_4 + 3x_5 + e_1$, $x_8 := 3x_6 + e_8$, $x_3 := -8x_4 - 5x_1 + e_3$, and $x_7 := -7x_1 + e_7$. (Here, all the $c_i$ are zero, but this is not the case in general.) The 'disturbance' variables $e_i$ are drawn mutually independently from *non-gaussian* distributions $p_i(e_i)$. Hidden variables are shown dashed; the observed data vector $\mathbf{x}$ consists of only the values of $x_1$, $x_2$, $x_3$, $x_5$, and $x_8$. **(b)** The canonical model corresponding to the network in (a). This is the observationally and causally equivalent network where all causally irrelevant variables have been simplified away. See main text for details.

any connections to it are not cause for concern. The reason for this is that the variable is not in any way relevant with respect to our goal: finding the causal relationships between the observed variables.

Another causally irrelevant hidden variable is $x_6$, which simply mediates the influence of $x_2$ and $x_4$ onto $x_8$. We have

$$\begin{aligned} x_6 &:= 2x_2 + 4x_4 + e_6 \\ x_8 &:= 3x_6 + e_8 \end{aligned}$$

leading to

$$\begin{aligned} x_8 &:= 3(2x_2 + 4x_4 + e_6) + e_8 \\ &= 6x_2 + 12x_4 + 3e_6 + e_8 \\ &= 6x_2 + 12x_4 + e_8' \end{aligned}$$

where we have simplified $e_8' = 3e_6 + e_8$. This shows that we can remove the hidden variable $x_6$ from the model to obtain a simpler model in which the observed data is identical to the original one and, in addition, the causal implications are the same. The resulting model is shown in Figure 1b. Note that hidden variable $x_4$ cannot be simplified away without changing the observed data.

We formalize this idea of causally relevant versus irrelevant hidden variables using the following concepts:

Two latent variable LiNGAM models are *observationally equivalent* if and only if the distribution $p(\mathbf{x})$ of the observed data vector is identical for the two models. This implies that the models cannot be distinguished based on observational data alone.

Two latent variable LiNGAM models are *observationally and causally equivalent* if and only if they are observationally equivalent and all causal effects of observed variables onto other observed variables are identical for the two models. In the notation of Pearl (2000) these causal effects are given by $p(\mathbf{x}_{J_1} \mid \mathrm{do}(\mathbf{x}_{J_2}))$, $J_1, J_2 \subseteq J$, with $J_1 \cap J_2 = \emptyset$. When these are identical for all choices of $J_1$ and $J_2$ the two models in question cannot be distinguished based on any observations nor any controlled experiments.

Finally, we define a *canonical model* to be any latent variable LiNGAM model where each latent variable is a root node (i.e. has no parents) and has at least two children (direct descendants). Furthermore, although different latent variables may have the same sets of children,

no two latent variables exhibit exactly proportional sets of connection strenghts to the observed variables. Finally, each latent variable is restricted to have zero mean and unit variance.

To derive a canonical model which is observationally and causally equivalent to any given latent variable LiNGAM model, we can use the following algorithm:

---

**Algorithm A:** Given a latent variable LiNGAM model, returns an observationally and causally equivalent canonical model

1. First remove any latent variables without children. Iterate this rule until there are no more such nodes.

2. For any connection of the form $X \to Y$, where $Y$ is a latent variable: (a) For all children $Z$ of $Y$, add a direct connection $X \to Z$, the strength of the connection being the product of the strenghts of $X \to Y$ and $Y \to Z$. If a direct connection already existed, add the specified amount to the original strength. (b) Remove the connection $X \to Y$. Iterate this rule until there are no more applicable connections.

3. Remove any latent variable with only a single child, incorporating the latent variable's disturbance variable and constant into those of the child. Iterate this until there are no more such latent variables.

4. For any pair of latent variables with *exactly proportional* sets of connection strenghts to the observed variables, combine these into a single latent variable. Iterate this until there are no more such pairs of latent variables.

5. Finally, standardize the latent variables to have zero mean and unit variance by adjusting the connection strenghts to, and the constants of, their children.

---

## 4 Model estimation

In this section we show how, from data generated by any stable latent variable LiNGAM model, to estimate the set of canonical models which are observationally equivalent to the generating model.

### 4.1 Model estimation by ICA

In earlier work (Shimizu et al., 2005; Hoyer et al., 2006) we showed that data generated from a LiNGAM process follows an independent component analysis (ICA) distribution (Comon, 1994; Hyvärinen et al., 2001). Here, we briefly review this concept, specifically focusing on the effect of latent variables.

We begin by considering the full data vector $\tilde{\mathbf{x}} = \{x_1, \ldots, x_m\}$, which includes the latent variables. If we as preprocessing subtract out the means of the variables, then the full data satisfies $\tilde{\mathbf{x}} = \tilde{\mathbf{B}}\tilde{\mathbf{x}} + \mathbf{e}$, where, because of the DAG assumption, $\tilde{\mathbf{B}}$ is a matrix that could be permuted to strict lower triangularity if one knew a causal ordering $k(i)$ of the variables. Solving for $\tilde{\mathbf{x}}$ one obtains $\tilde{\mathbf{x}} = \tilde{\mathbf{A}}\mathbf{e}$, where $\tilde{\mathbf{A}} = (\mathbf{I} - \tilde{\mathbf{B}})^{-1}$ contains the influence of the disturbance variables onto the observed variables. Again, $\tilde{\mathbf{A}}$ could be permuted to lower triangularity (although not *strict* lower triangularity) with an appropriate permutation $k(i)$. Taken together, the linear relationship between $\mathbf{e}$ and $\tilde{\mathbf{x}}$ and the independence and non-gaussianity of the components of $\mathbf{e}$ define the standard linear *independent component analysis* model.

So far, this is just restating what we pointed out in our previous work. Now consider the effect of hiding some of the variables. This yields $\mathbf{x} = \mathbf{A}\mathbf{e}$, where $\mathbf{A}$ contains just the rows of $\tilde{\mathbf{A}}$ corresponding to the observed variables. When the number of observed variables is less than the number of disturbance variables, $\mathbf{A}$ is non-square with more columns than rows. This is known as an *overcomplete* basis in the ICA literature.

Let us take a closer look at the structure inherent in the 'mixing matrix' $\mathbf{A}$. First, note that since for every latent variable LiNGAM model there is an observationally and causally equivalent canonical model, we can without loss of generality restrict our analysis to canonical models. Next, arrange the full set of variables such that all latent variables come first (in any

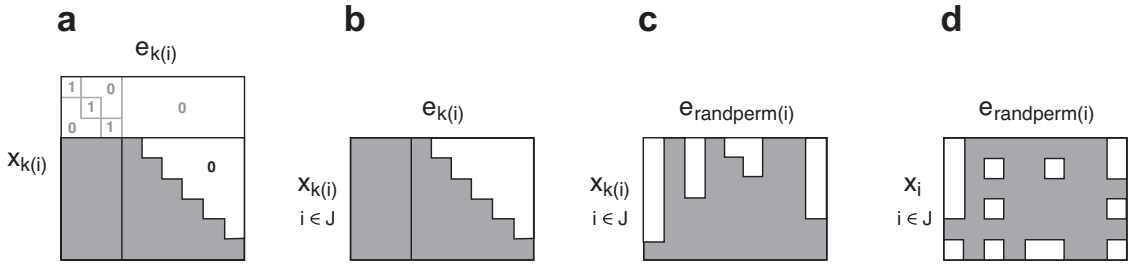



Figure 2: **(a)** Basic structure of the full ICA matrix $\widetilde{\mathbf{A}}$ for a canonical model. (In this example, there are 3 hidden and 6 observed variables.) The top rows correspond to the hidden variables. Note that the assumption of a canonical model implies that this part of the matrix consists of an identity submatrix and zeros. The bottom rows correspond to the observed variables. Here, the shaded area represents values which, depending on the network, *may be zero or non-zero*. The white area represents entries which are zero by the DAG-assumption. **(b)** Since, by definition, we do not observe the latent variables, the information that we have is limited to the bottom part of the matrix shown in (a). **(c)** Due to the permutation indeterminancy in ICA, the observed basis matrix has its columns in random order. **(d)** Since we do not (up front) know a causal order for the observed variables, the observed mixing matrix $\mathbf{A}$ has the rows in the order of data input, not in a causal order.

internal order) followed by all observed variables (in a causal order), and look at the structure of the full matrix $\widetilde{\mathbf{A}}$ shown in Figure 2a. Although we only observe part of the full matrix, with randomly permuted columns and with arbitrarily permuted rows as in Figure 2d, the crucial point is that the observed ICA basis matrix $\mathbf{A}$ contains all of the information contained in the full matrix $\widetilde{\mathbf{A}}$ in the sense that all the free parameters in $\widetilde{\mathbf{A}}$ are also contained in $\mathbf{A}$. Thus there is hope that the causal model could be reconstructed from the observed basis matrix $\mathbf{A}$.

At this point, we note that we of course do not directly observe $\mathbf{A}$ but must infer it from the sample vectors. Eriksson and Koivunen (2004) have recently shown that the overcomplete ICA model is identifiable given enough data, and several algorithms are available for this task, see e.g. (Moulines et al., 1997; Attias, 1999; Hyvärinen et al., 2001). Thus, the remainder of this subsection considers the inference of the causal model were the exact ICA mixing matrix $\mathbf{A}$ known. The next subsection deals with the practical aspect of dealing with inevitable estimation errors.

As in the standard square ICA case, identification in the overcomplete case is only up to permutation and scaling of the columns of $\mathbf{A}$. The scaling indeterminancy is not serious; it simply amounts to a problem of not being able to attribute the magnitude of the influence of a disturbance variable $e_i$ to its variance, the strength of its connection to its corresponding variable $x_i$, and in the case of hidden variables the average strength of the connections from that hidden variable to the observed variables. This is of no consequence since we are anyway never able to directly monitor the hidden variables nor the disturbance variables, making the scaling simply a matter of definition.

In contrast, the permutation indeterminancy *is* a serious problem, and in general leads to non-uniqueness in inferring the model: We cannot know which columns of $\mathbf{A}$ correspond to the hidden variables. Note, however, that this is the only information missing, as illustrated in Figure 2. Thus, an upper bound for the number of observationally equivalent canonical models is the number of classifications into observed vs hidden. This is simply $(N_o + N_h)!/(N_o!N_h!)$, where $N_o$ and $N_h$ denote the numbers of observed and hidden variables. For each classification we need to check whether this leads to a valid latent-variable LiNGAM model:

**Algorithm B:** Given an overcomplete basis $\mathbf{A}$ (containing exact zeros) and the means of the observed variables, calculates all observationally equivalent canonical latent variable LiNGAM models compatible with the basis

1. $N_h$ is determined as the number of columns of $\mathbf{A}$ minus the number of rows.
2. For each possible classification of the columns of $\mathbf{A}$ as belonging to disturbance variables of observed vs hidden variables:
   (a) Reorder the columns such that the ones selected as 'hidden variables' come first
   (b) Augment the basis by adding the unobserved top part of the matrix (as in Figure 2a), obtaining an estimate of $\widetilde{\mathbf{A}}$.
   (c) Test whether it is possible to permute $\widetilde{\mathbf{A}}$ (using independent row and column permutations) to lower-triangular form. If not, go to the next classification.
   (d) Divide each column of $\widetilde{\mathbf{A}}$ by its diagonal element, and calculate the connection strengths $\widetilde{\mathbf{B}} = \mathbf{I} - \widetilde{\mathbf{A}}^{-1}$
   (e) Check that the found network is compatible with the *stability* assumption. If not, go to the next classification.
   (f) Add the found network $\widetilde{\mathbf{B}}$ to the list of observationally equivalent models which could have generated the data.

### 4.2 Practical considerations

Up to here we have assumed that the exact ICA basis matrix is known. In a real implementation, of course, it can only be *estimated* from the data, inevitably leading to (small) estimation errors, causing all elements of $\mathbf{A}$ to be non-zero and making Algorithm B not directly applicable. Furthermore, since the structure of the network is related to the *inverse* of $\mathbf{A}$, small estimation errors will give rise to small direct effects where none exist in the generating model.

Solving these problems requires us to have an idea of the accuracy of our estimate of $\mathbf{A}$. Here, we advocate using resampling methods (Efron and Tibshirani, 1993), obtaining a *set* of estimates $\mathbf{A}_i$ representing our uncertainty regarding the elements of the mixing matrix. This set can then be used as follows: First, one can infer which elements of $\mathbf{A}$ are exactly zero by standard statistical testing, taking as the null hypothesis the assumption that a given element is zero and rejecting it when the mean/variance ratio for that element is large enough (in an absolute value sense). This procedure will give the correct answer with probability approaching 1 as the amount of data grows.

Having identified the zeros of the basis matrix, we apply Algorithm B to each estimate $\mathbf{A}_i$ to yield a set of estimates for the generating causal model.[1] This set can, again, be utilized in statistical tests to prune out the small direct effects which result from estimation errors.

A full description of the above ideas is out of the scope of this short paper. Nevertheless, they have been successfully implemented in our simulations (see Section 5) and the reader is invited to study the code package for all the details.

## 5 Simulations

We have performed extensive simulations in order to (i) verify the algorithms described, (ii) test the viability of the resampling approach for dealing with estimation errors, and (iii) provide a simple demonstration of learning a small hidden variable LiNGAM model from data. Full well-documented Matlab code[2] for all of these experiments is available, to allow the reader to effortlessly replicate and verify our results, which unfortunately cannot be described in very great detail here due to lack of space.

First, we tested the basic idea by generating random latent variable LiNGAM models and, for each model, calculating its corresponding ICA basis matrix $\mathbf{A}$ and from it inferring the set of observationally equivalent canonical models using Algorithm B. This process is illustrated

[1] Note that this is, in general, a set of sets: One index corresponds to the different possible permutations of the columns of $\mathbf{A}$, the other to the different estimates by resampling.

[2] at `http://www.cs.helsinki.fi/patrik.hoyer/code/lvlingam.tar.gz`

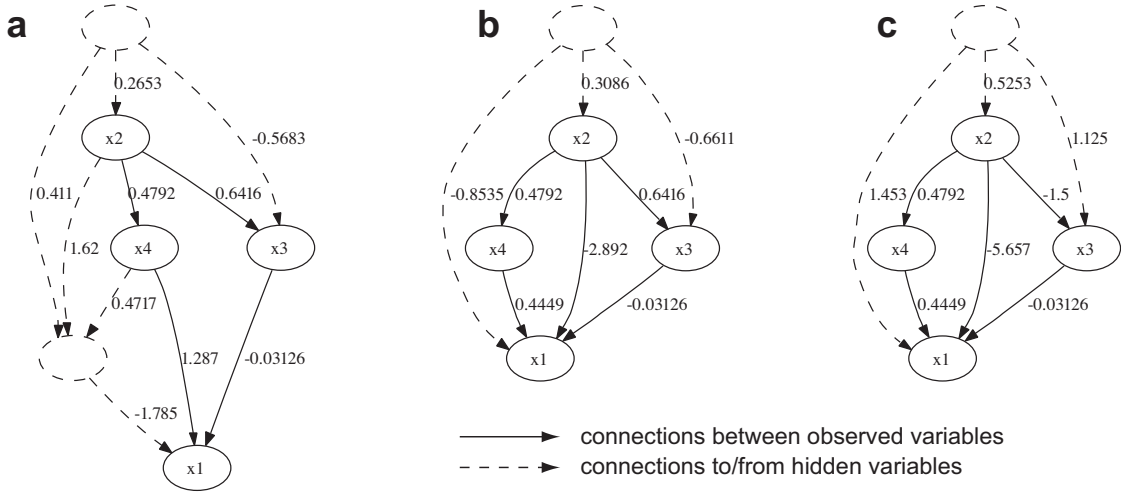



Figure 3: **(a)** A randomly generated latent variable LiNGAM model. **(b)** The canonical model which is causally and observationally equivalent to the network in (a). **(c)** An observationally equivalent (but causally *not* equivalent) model. From the ICA basis which model (a) generates, it is impossible to distinguish between models (b) and (c). See main text for details.

in Figure 3. In every case, exactly one model in the inferred set was causally equivalent to the original model.

Second, we tested the practical viability of the approach by assuming that we only have a set of inexact estimates of the basis matrix: We generated random LiNGAM models and, for each model, calculated its corresponding ICA basis matrix, added noise to yield 20 different 'estimates' of it, and used the approach of Section 4.2 to infer the set of possible generating models. In each case, one of the inferred models was close (in a causal sense) to the original model. When the noise becomes large enough to cause misclassifications of zeros, the method can fail and the resulting networks are not necessarily close to the generating one. This is analogous to the sensitivity of constraint-based causal discovery algorithms (Pearl, 2000; Spirtes et al., 2000) to misclassifications of conditional independencies in the observed data.

Finally, we performed a small-scale demonstration of learning the model from actual data. We used the mixture-of-gaussians framework (Moulines et al., 1997; Attias, 1999) to estimate the ICA bases. To make the problem as simple as possible, the simulation was done on very small networks (3 observed and one hidden variable) and we additionally assumed that we knew the distributions of all the disturbance variables. A sample of 1000 data vectors was used. Figure 4 shows a typical result. The algorithm here finds the correct structure of the network (in this case the network is uniquely identifiable). This would not be possible with conventional methods as there are no conditional independencies among the observed variables.

## 6 Future work

Much work remains before the framework described here can be applied to practical data analysis problems. The most formidable problem is that of reliably estimating an overcomplete ICA basis matrix when the number of hidden variables and the distributions of the disturbance variables are unknown. Current algorithms (Moulines et al., 1997; Attias, 1999) may work for very small dimensions but to our knowledge no methods are yet available for estimating overcomplete ICA bases reliably and accurately in high dimensions.

Fortunately, we may be able to solve relatively large problems even with current methods. The key is that hidden variables only cause what might be called *local overcompleteness*. In

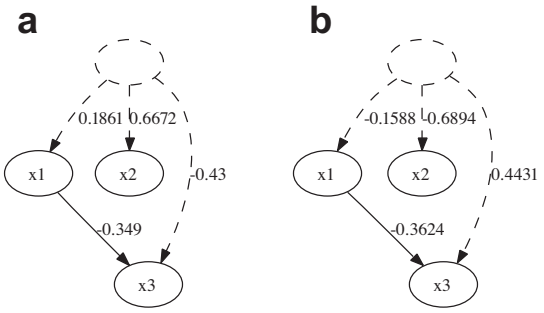



Figure 4: **(a)** A randomly generated latent variable LiNGAM model used to generate data. **(b)** The model estimated from the data. Note that we cannot determine the sign of the influence of the hidden variable since its distribution was symmetric. All other parameters are correctly identified, including the absence of connections between $x_1$ and $x_2$, and $x_2$ and $x_3$.

a large LiNGAM model where each latent variable directly influences only a small number of observed variables, the data follows an independent subspace (ISA) model (Hyvärinen and Hoyer, 2000). In such a model, the data distribution can be represented as a combination of low-dimensional independent subspaces. When the data follows the ISA model, the individual subspaces might still be found using efficient ICA algorithms (Hyvärinen et al., 2001), and algorithms for overcomplete ICA would only need to be run *inside each subspace*.

## 7 Conclusions

We have recently shown how to estimate linear causal models when the distributions involved are non-gaussian and no confounding hidden variables exist (Shimizu et al., 2005; Hoyer et al., 2006). In this contribution, we have described the effects of confounding latent variables, and shown how to estimate the set of models consistent with the observed data. Simulations, for which full Matlab code is available, confirm the viability of the approach. Further work is needed to develop the software into a practical, easy-to-use data-analysis tool.

### Acknowledgments

We thank Aapo Hyvärinen for comments on the manuscript. P.O.H. was supported by the Academy of Finland project #204826.

## References

H. Attias. 1999. Independent factor analysis. *Neural Computation*, 11:803–851.

K. A. Bollen. 1989. *Structural Equations with Latent Variables*. John Wiley & Sons.

P. Comon. 1994. Independent component analysis – a new concept? *Signal Processing*, 36:287–314.

B. Efron and R. Tibshirani. 1993. *Introduction to the Boostrap*. Chapman and Hall.

J. Eriksson and V. Koivunen. 2004. Identifiability, separability and uniqueness of linear ICA models. *IEEE Signal Processing Letters*, 11(7):601–604.

P. O. Hoyer, S. Shimizu, A. Hyvärinen, Y. Kano, and A. J. Kerminen. 2006. New permutation algorithms for causal discovery using ICA. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2006)*, pages 115–122.

A. Hyvärinen and P. O. Hoyer. 2000. Emergence of phase and shift invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, 12(7):1705–1720.

A. Hyvärinen, J. Karhunen, and E. Oja. 2001. *Independent Component Analysis*. Wiley Interscience.

E. Moulines, J.-F. Cardoso, and E. Gassiat. 1997. Maximum likelihood for blind separation and deconvolution of noisy signals using mixture models. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'97)*, pages 3617–3620, Munich, Germany.

J. Pearl. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

S. Shimizu, A. Hyvärinen, Y. Kano, and P. O. Hoyer. 2005. Discovery of non-gaussian linear causal models using ICA. In *Proc. 21st Conference on Uncertainty in Artificial Intelligence (UAI-2005)*, pages 526–533, Edinburgh, Scotland.

R. Silva, R. Scheines, C. Glymour, and P. Spirtes. 2006. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7:191–246.

P. Spirtes, C. Glymour, and R. Scheines. 2000. *Causation, Prediction, and Search*. MIT Press.

# Symmetric Causal Independence Models for Classification

Rasa Jurgelenaite and Tom Heskes
Institute for Computing and Information Sciences
Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

**Abstract**

Causal independence modelling is a well-known method both for reducing the size of probability tables and for explaining the underlying mechanisms in Bayesian networks. In this paper, we propose an application of an extended class of causal independence models, causal independence models based on the symmetric Boolean function, for classification. We present an EM algorithm to learn the parameters of these models, and study convergence of the algorithm. Experimental results on the Reuters data collection show the competitive classification performance of causal independence models based on the symmetric Boolean function in comparison to noisy OR model and, consequently, with other state-of-the-art classifiers.

## 1 Introduction

*Bayesian networks* (Pearl, 1988) are well-established as a sound formalism for representing and reasoning with probabilistic knowledge. However, because the number of conditional probabilities for the node grows exponentially with the number of its parents, it is usually unreliable if not infeasible to specify the conditional probabilities for the node that has a large number number of parents. The task of assessing conditional probability distributions becomes even more complex if the model has to integrate expert knowledge. While learning algorithms can be forced to take into account an expert's view, for the best possible results the experts must be willing to reconsider their ideas in light of the model's 'discovered' structure. This requires a clear understanding of the model by the domain expert. *Causal independence models* (Díez, 1993; Heckerman and Breese, 1994; Srinivas, 1993; Zhang and Poole, 1996) can both limit the number of conditional probabilities to be assessed and provide the ability for models to be understood by domain experts in the field. The main idea of causal independence models is that causes influence a given common effect through intermediate variables and interaction function.

Causal independence assumptions are often used in practical Bayesian network models (Kappen and Neijt, 2002; Shwe et al., 1991). However, most researchers restrict themselves to using only the logical OR and logical AND operators to define the interaction among causes. The resulting probabilistic submodels are called *noisy OR* and *noisy AND*; their underlying assumption is that the presence of either at least one cause or all causes at the same time give rise to the effect. Several authors proposed to expand the space of interaction functions by other symmetric Boolean functions: the idea was already mentioned but not developed further in (Meek and Heckerman, 1997), analysis of the qualitative patterns was presented in (Lucas, 2005), and assessment of conditional probabilities was studied in (Jurgelenaite et al., 2006).

Even though for some real-world problems the intermediate variables are observable (see Visscher et al. (2005)), in many problems these variables are latent. Therefore, conditional probability distributions depend on unknown parameters which must be estimated from data,

using *maximum likelihood* (ML) or *maximum a posteriori* (MAP). One of the most widespread techniques for finding ML or MAP estimates is the *expectation-maximization* (EM) algorithm. Meek and Heckerman (1997) provided a general scheme how to use the EM algorithm to compute the maximum likelihood estimates of the parameters in causal independence models assumed that each local distribution function is collection of multinomial distributions. Vomlel (2006) described the application of the EM algorithm to learn the parameters in the noisy OR model for classification.

The application of an extended class of causal independence models, namely causal independence models with a symmetric Boolean function as an interaction function, to classification is the main topic of this paper. These models will further be referred to as the *symmetric causal independence models*. We present an EM algorithm to learn the parameters in symmetric causal independence models, and study convergence of the algorithm. Experimental results show the competitive classification performance of the symmetric causal independence models in comparison with the noisy OR classifier and, consequently, with other widely-used classifiers.

The remainder of this paper is organised as follows. In the following section, we review Bayesian networks and discuss the semantics of symmetric causal independence models. In Section 3, we state the EM algorithm for finding the parameters in symmetric causal independence models. The maxima of the log-likelihood function for the symmetric causal independence models are examined in Section 4. Finally, Section 5 presents the experimental results, and conclusions are drawn in Section 6.

## 2 Symmetric Boolean Functions for Modelling Causal Independence

### 2.1 Bayesian Networks

A *Bayesian network* $\mathcal{B} = (G, \Pr)$ represents a factorised joint probability distribution on a set of random variables $\mathbf{V}$. It consists of two parts: (1) a qualitative part, represented as an acyclic directed graph (ADG) $G = (\mathbf{V}(G), \mathbf{A}(G))$,

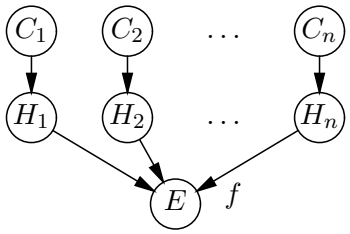



Figure 1: Causal independence model

where there is a 1–1 correspondence between the vertices $\mathbf{V}(G)$ and the random variables in $\mathbf{V}$, and arcs $\mathbf{A}(G)$ represent the conditional (in)dependencies between the variables; (2) a quantitative part Pr consisting of local probability distributions $\Pr(V \mid \pi(V))$, for each variable $V \in \mathbf{V}$ given the parents $\pi(V)$ of the corresponding vertex (interpreted as variables). The joint probability distribution Pr is factorised according to the structure of the graph, as follows:

$$\Pr(\mathbf{V}) = \prod_{V \in \mathbf{V}} \Pr(V \mid \pi(V)).$$

Each variable $V \in \mathbf{V}$ has a finite set of mutually exclusive states. In this paper, we assume all variables to be binary; as an abbreviation, we will often use $v^+$ to denote $V = \top$ (true) and $v^-$ to denote $V = \bot$ (false). We interpret $\top$ as 1 and $\bot$ as 0 in an arithmetic context. An expression such as

$$\sum_{\psi(H_1,\ldots,H_n)=\top} g(H_1, \ldots, H_n)$$

stands for summing $g(H_1, \ldots, H_n)$ over all possible values of the variables $H_k$ for which the constraint $\psi(H_1, \ldots, H_n) = \top$ holds.

### 2.2 Semantics of Symmetric Causal Independence Models

Causal independence is a popular way to specify interactions among cause variables. The global structure of a causal independence model is shown in Figure 1; it expresses the idea that causes $C_1, \ldots, C_n$ influence a given common effect $E$ through hidden variables $H_1, \ldots, H_n$ and a deterministic function $f$, called the *interaction function*. The impact of each cause $C_i$ on the common effect $E$ is independent of each

other cause $C_j, j \neq i$. The hidden variable $H_i$ is considered to be a contribution of the cause variable $C_i$ to the common effect $E$. The function $f$ represents in which way the hidden effects $H_i$, and indirectly also the causes $C_i$, interact to yield the final effect $E$. Hence, the function $f$ is defined in such a way that when a relationship, as modelled by the function $f$, between $H_i, i = 1, \ldots, n$, and $E = \top$ is satisfied, then it holds that $f(H_1, \ldots, H_n) = \top$. It is assumed that $\Pr(e^+ \mid H_1, \ldots, H_n) = 1$ if $f(H_1, \ldots, H_n) = \top$, and $\Pr(e^+ \mid H_1, \ldots, H_n) = 0$ if $f(H_1, \ldots, H_n) = \bot$.

A causal independence model is defined in terms of the causal parameters $\Pr(H_i \mid C_i)$, for $i = 1, \ldots, n$ and the function $f(H_1, \ldots, H_n)$. Most papers on causal independence models assume that absent causes do not contribute to the effect (Heckerman and Breese, 1994; Pearl, 1988). In terms of probability theory this implies that it holds that $\Pr(h_i^+ \mid c_i^-) = 0$; as a consequence, it holds that $\Pr(h_i^- \mid c_i^-) = 1$. In this paper we make the same assumption.

In situations in which the model does not capture all possible causes, it is useful to introduce a *leaky cause* which summarizes the unidentified causes contributing to the effect and is assumed to be always present (Henrion, 1989). We model this leak term by adding an additional input $C_{n+1} = 1$ to the data; in an arithmetic context the leaky cause is treated in the same way as identified causes.

The conditional probability of the occurrence of the effect $E$ given the causes $C_1, \ldots, C_n$, i.e., $\Pr(e^+ \mid C_1, \ldots, C_n)$, can be obtained from the causal parameters $\Pr(H_l \mid C_l)$ as follows (Zhang and Poole, 1996):

$$\Pr(e^+ \mid C_1, \ldots, C_n) = \sum_{f(H_1,\ldots,H_n)=\top} \prod_{i=1}^{n} \Pr(H_i \mid C_i). \quad (1)$$

In this paper, we assume that the function $f$ in Equation (1) is a Boolean function. However, there are $2^{2^n}$ different $n$-ary Boolean functions (Enderton, 1972; Wegener, 1987); thus, the potential number of causal interaction models is huge. However, if we assume that the order of the cause variables does not matter, the Boolean functions become *symmetric* (Wegener, 1987) and the number reduces to $2^{n+1}$.

An important symmetric Boolean function is the *exact* Boolean function $\epsilon_l$, which has function value true, i.e. $\epsilon_l(H_1, \ldots, H_n) = \top$, if $\sum_{i=1}^{n} \nu(H_i) = l$ with $\nu(H_i)$ equal to 1, if $H_i$ is equal to true and 0 otherwise. A symmetric Boolean function can be decomposed in terms of the exact functions $\epsilon_l$ as (Wegener, 1987):

$$f(H_1, \ldots, H_n) = \bigvee_{i=0}^{n} \epsilon_i(H_1, \ldots, H_n) \wedge \gamma_i \quad (2)$$

where $\gamma_i$ are Boolean constants depending only on the function $f$. For example, for the Boolean function defined in terms of the OR operator we have $\gamma_0 = \bot$ and $\gamma_1 = \ldots = \gamma_n = \top$.

Another useful symmetric Boolean function is the *threshold* function $\tau_k$, which simply checks whether there are at least $k$ trues among the arguments, i.e. $\tau_k(I_1, \ldots, I_n) = \top$, if $\sum_{j=1}^{n} \nu(I_j) \geq k$ with $\nu(I_j)$ equal to 1, if $I_j$ is equal to true and 0 otherwise. To express it in the Boolean constants we have: $\gamma_0 = \quad = \gamma_{k-1} = \bot$ and $\gamma_k = \quad = \gamma_n = \top$. Causal independence model based on the Boolean threshold function further will be referred to as the *noisy threshold models*.

### 2.3 The Poisson Binomial Distribution

Using the property of Equation (2) of the symmetric Boolean functions, the conditional probability of the occurrence of the effect $E$ given the causes $C_1, \ldots, C_n$ can be decomposed in terms of probabilities that exactly $l$ hidden variables $H_1, \ldots, H_n$ are true as follows:

$$\Pr(e^+ \mid C_1, \ldots, C_n) = \sum_{\substack{0 \leq l \leq n \\ \gamma_l}} \sum_{\epsilon_l(H_1,\ldots,H_n)} \prod_{i=1}^{n} \Pr(H_i \mid C_i).$$

Let $l$ denote the number of successes in $n$ independent trials, where $p_i$ is a probability of success in the $i$th trial, $i = 1, \ldots, n$; let $\mathbf{p} = (p_1, \ldots, p_n)$, then $\mathrm{B}(l; \mathbf{p})$ denotes the *Poisson binomial distribution* (Le Cam, 1960; Dar-

roch, 1964):

$$\mathrm{B}(l;\mathbf{p}) = \prod_{i=1}^{n}(1-p_i)\sum_{1\le j_1<\ldots<j_l\le n}\prod_{z=1}^{l}\frac{p_{j_z}}{1-p_{j_z}}.$$

Let us define a vector of probabilistic parameters $\mathbf{p}(C_1,\ldots,C_n) = (p_1,\ldots,p_n)$ with $p_i = \Pr(h_i^+ \mid C_i)$. Then the connection between the Poisson binomial distribution and the class of symmetric causal independence models is as follows.

**Proposition 1.** *It holds that:*

$$\Pr(e^+ \mid C_1,\ldots,C_n) = \sum_{i=0}^{n}\mathrm{B}(i;\mathbf{p}(C_1,\ldots,C_n))\gamma_i.$$

## 3 EM Algorithm

Let $\mathbf{D} = \{\mathbf{x}^1,\ldots,\mathbf{x}^N\}$ be a data set of independent and identically distributed settings of the observed variables in a symmetric causal independence model, where

$$\mathbf{x}^j = (\mathbf{c}^j, e^j) = (c_1^j,\ldots,c_n^j,e^j).$$

We assume that no additional information about the model is available. Therefore, to learn the parameters of the model we maximize the conditional log-likelihood

$$CLL(\boldsymbol{\theta}) = \sum_{j=1}^{N}\ln\Pr(e^j \mid \mathbf{c}^j,\boldsymbol{\theta}).$$

The expectation-maximization (EM) algorithm (Dempster et al., 1977) is a general method to find the maximum likelihood estimate of the parameters in probabilistic models, where the data is incomplete or the model has hidden variables.

Let $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_n)$ be the parameters of the symmetric causal independence model where $\theta_i = \Pr(h_i^+ \mid c_i^+)$. Then, after some calculations, the $(z+1)$-th iteration of the EM algorithm for symmetric causal independence models is given by:

**Expectation step**: For every data sample $\mathbf{x}^j = (\mathbf{c}^j, e^j)$ with $j = 1,\ldots,N$, we form

$$\mathbf{p}^{(z,j)} = (p_1^{(z,j)},\ldots,p_n^{(z,j)}) \text{ where } p_i^{(z,j)} = \theta_i^{(z)}c_i^j.$$

Let us define

$$\mathbf{p}_{\setminus k}^{(z,j)} = (p_1^{(z,j)},\ldots,p_{k-1}^{(z,j)},p_{k+1}^{(z,j)},\ldots,p_n^{(z,j)}).$$

Subsequently, for all hidden variables $H_k$ with $k = 1,\ldots,n$ we compute the probability $\Pr(h_k^+ \mid e^j,\mathbf{c}^j,\boldsymbol{\theta}^{(z)})$ where

$$\Pr(h_k^+ \mid e^j,\mathbf{c}^j,\boldsymbol{\theta}^{(z)}) = \frac{p_k^{(z,j)}\sum_{i=0}^{n-1}\mathrm{B}\left(i;\mathbf{p}_{\setminus k}^{(z,j)}\right)\gamma_{i+1}}{\sum_{i=0}^{n}\mathrm{B}\left(i;\mathbf{p}^{(z,j)}\right)\gamma_i} \text{ if } e^j = 1,$$

and

$$\Pr(h_k^+ \mid e^j,\mathbf{c}^j,\boldsymbol{\theta}^{(z)}) = \frac{p_k^{(z,j)}\left(1-\sum_{i=0}^{n-1}\mathrm{B}\left(i;\mathbf{p}_{\setminus k}^{(z,j)}\right)\gamma_{i+1}\right)}{1-\sum_{i=0}^{n}\mathrm{B}\left(i;\mathbf{p}^{(z,j)}\right)\gamma_i} \text{ if } e^j = 0.$$

**Maximization step**: Update the parameter estimates for all $k = 1,\ldots,n$:

$$\theta_k = \frac{\sum_{1\le j\le N}c_k^j\Pr(h_k^+ \mid e^j,\mathbf{c}^j,\boldsymbol{\theta}^{(z)})}{\sum_{1\le j\le N}c_k^j}.$$

## 4 Analysis of the Maxima of the Log-likelihood Function

Generally, there is no guarantee that the EM algorithm will converge to a global maximum of log-likelihood. In this section, we investigate the maxima of the conditional log-likelihood function for symmetric causal independence models.

### 4.1 Noisy OR and Noisy AND Models

In this section we will show that the conditional log-likelihood for the noisy OR and the noisy AND models has only one maximum. Since the conditional log-likelihood for these models is not necessarily concave we will use a monotonic transformation to prove the absence of the stationary points other than global maxima.

First, we establish a connection between the maxima of the log-likelihood function and the maxima of the corresponding composite function.

**Proposition 2.** ***(Global optimality condition for concave functions (Boyd and Vandenberghe, 2004))***
*Suppose* $h(\mathbf{q}) : \mathbf{Q} \rightarrow \Re$ *is concave and differentiable on* $\mathbf{Q}$. *Then* $\mathbf{q}^* \in \mathbf{Q}$ *is a global maximum if and only if*

$$\nabla h(\mathbf{q}^*) = \left( \frac{\partial h(\mathbf{q}^*)}{\partial q_1}, \ldots, \frac{\partial h(\mathbf{q}^*)}{\partial q_n} \right)^T = \mathbf{0}.$$

Further we consider the function

$$CLL(\boldsymbol{\theta}) = h(\mathbf{q}(\boldsymbol{\theta})).$$

Let $CLL(\boldsymbol{\theta})$ and $h(\mathbf{q}(\boldsymbol{\theta}))$ be twice differentiable functions, and let $\mathbf{q}(\boldsymbol{\theta})$ be a differentiable, injective function where $\boldsymbol{\theta}(\mathbf{q})$ is its inverse. Then the following relationship between the stationary points of the functions $CLL$ and $h$ holds.

**Lemma 1.** *Suppose,* $\boldsymbol{\theta}^*$ *is a stationary point of* $CLL(\boldsymbol{\theta})$. *Then there is a corresponding point* $\mathbf{q}(\boldsymbol{\theta}^*)$, *which is a stationary point of* $h(\mathbf{q}(\boldsymbol{\theta}))$.

*Proof.* Since the function $\mathbf{q}(\boldsymbol{\theta})$ is differentiable and injective, its Jacobian matrix $\frac{\partial(q_1,\ldots,q_n)}{\partial(\theta_1,\ldots,\theta_n)}$ is positive definite. Therefore, from the chain rule it follows that if $\nabla CLL(\boldsymbol{\theta}^*) = \mathbf{0}$, then $\nabla h(\mathbf{q}(\boldsymbol{\theta}^*)) = \mathbf{0}$. □

**Proposition 3.** *If* $h(\mathbf{q}(\boldsymbol{\theta}))$ *is concave and* $\boldsymbol{\theta}^*$ *is a stationary point of* $CLL(\boldsymbol{\theta})$, *then* $\boldsymbol{\theta}^*$ *is a global maximum.*

*Proof.* If $\boldsymbol{\theta}^*$ is a stationary point, then from Lemma 1 it follows that $\mathbf{q}(\boldsymbol{\theta}^*)$ is also stationary. From the global optimality condition for concave functions the stationary point $\mathbf{q}(\boldsymbol{\theta}^*)$ is a maximum of $h(\mathbf{q}(\boldsymbol{\theta}))$, thus from the definition of global maximum we get that for all $\boldsymbol{\theta}$

$$CLL(\boldsymbol{\theta}) = h(\mathbf{q}(\boldsymbol{\theta})) \leq h(\mathbf{q}(\boldsymbol{\theta}^*)) = CLL(\boldsymbol{\theta}^*).$$

□

Given Proposition 3 the absence of the local optima can be proven by introducing such a monotonic transformation $\mathbf{q}(\boldsymbol{\theta})$ that the composite function $h(\mathbf{q}(\boldsymbol{\theta}))$ would be concave. As it is a known result that the Hessian matrix of the log-likelihood function for logistic regression is negative-semidefinite, and hence the problem has no local optima, we will use transformations that allow us to write the log-likelihood for the noisy OR and noisy AND models in a similar form as that of the logistic regression model.

The conditional probability of the effect in a noisy OR model can be written:

$$\Pr(e^+ \mid \mathbf{c}, \boldsymbol{\theta}) = 1 - \prod_{i=1}^{n} \Pr(h_i^- \mid c_i)$$
$$= 1 - \prod_{i=1}^{n} (1-\theta_i)^{c_i} = 1 - \exp\left( \sum_{i=1}^{n} \ln(1-\theta_i) c_i \right).$$

Let us choose a monotonic transformation $q_i = -\ln(1-\theta_i), i = 1, \ldots, n$. Then the conditional probability of the effect in a noisy OR model equals

$$\Pr(e^+ \mid \mathbf{c}, \mathbf{q}) = 1 - e^{-\mathbf{q}^T \mathbf{c}}.$$

Let us define $z^j = \mathbf{q}^T \mathbf{c}^j$ and $f(z^j) = \Pr(e^+ \mid \mathbf{c}^j, \mathbf{q})$, then the function $h$ reads

$$h(\mathbf{q}) = \sum_{j=1}^{N} e^j \ln f(z^j) + (1 - e^j) \ln(1 - f(z^j)). \quad (3)$$

Since $f'(z^j) = 1 - f(z^j)$, the first derivative of $h$ is

$$\frac{\partial h(\mathbf{q})}{\partial \mathbf{q}} = \sum_{j=1}^{N} \frac{f'(z^j)(e^j - f(z^j))}{f(z^j)(1 - f(z^j))} \mathbf{c}^j$$
$$= \sum_{j=1}^{N} \frac{e^j - f(z^j)}{f(z^j)} \mathbf{c}^j.$$

To prove that the function $h$ is concave we need to prove that its Hessian matrix is negative semidefinite. The Hessian matrix of $h$ reads

$$\frac{\partial^2 h(\mathbf{q})}{\partial \mathbf{q} \partial \mathbf{q}^T} = -\sum_{j=1}^{N} \frac{1 - f(z^j)}{f(z^j)^2} e^j \mathbf{c}^j \mathbf{c}^{j\,T}.$$

As the Hessian matrix of $h$ is negative semidefinite, the function $h$ is concave. Therefore, from Proposition 3 it follows that every stationary point of the log-likelihood function for the noisy OR model is a global maximum.

The conditional probability of the effect in a noisy AND model can be written:

$$\begin{aligned}\Pr(e^{j+} \mid \mathbf{c}, \boldsymbol{\theta}) &= \prod_{i=1}^{n} \Pr(h_i^+ \mid c_i) \\ &= \prod_{i=1}^{n} \theta_i^{c_i} = \exp\left(\sum_{i=1}^{n} \ln \theta_i c_i\right).\end{aligned}$$

Let us choose a monotonic transformation $q_i = \ln \theta_i, i = 1, \ldots, n$. Then the conditional probability of the effect in a noisy AND model equals

$$\Pr(e^{j+} \mid \mathbf{c}, \mathbf{q}) = e^{\mathbf{q}^T \mathbf{c}}.$$

Let us define $z^j = \mathbf{q}^T \mathbf{c}^j$ and $f(z^j) = \Pr(e^+ \mid \mathbf{c}^j, \mathbf{q})$. The function $h$ is the same as for the noisy OR model in Equation (3). Combined with $f'(z^j) = f(z^j)$, it yields the first derivative of $h$

$$\begin{aligned}\frac{\partial h(\mathbf{q})}{\partial \mathbf{q}} &= \sum_{j=1}^{N} \frac{f'(z^j)(e^j - f(z^j))}{f(z^j)(1 - f(z^j))} \mathbf{c}^j \\ &= \sum_{j=1}^{N} \frac{e^j - f(z^j)}{1 - f(z^j)} \mathbf{c}^j\end{aligned}$$

and Hessian matrix

$$\frac{\partial^2 h(\mathbf{q})}{\partial \mathbf{q} \partial \mathbf{q}^T} = -\sum_{j=1}^{N} \frac{f(z^j)}{(1 - f(z^j))^2}(1 - e^j)\mathbf{c}^j \mathbf{c}^{j\,T}.$$

Hence, the function $h$ is concave, and the log-likelihood for the noisy AND model has no other stationary points than the global maxima.

### 4.2 General Case

The EM algorithm is guaranteed to converge to the local maxima or saddle points. Thus, we can only be sure that the global maximum, i.e. a point $\boldsymbol{\theta}^*$ such that $CLL(\boldsymbol{\theta}^*) \geq CLL(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}^* \neq \boldsymbol{\theta}$, will be found if the log-likelihood has neither saddle points nor local maxima. However, the log-likelihood function for a causal independence model with any symmetric Boolean function does not always fulfill this requirement as it is shown in the following counterexample.

**Example 1.** Let us assume a data set $\mathbf{D} = \{(1,1,1,1),(1,0,1,0)\}$ and an interaction function $\epsilon_1$, i.e. $\gamma_1 = 1$ and $\gamma_0 = \gamma_2 = \gamma_3 = 0$. To learn the hidden parameters of the model describing this interaction we have to maximize the conditional log-likelihood function

$$\begin{aligned}CLL(\boldsymbol{\theta}) = \ln[&\theta_1(1-\theta_2)(1-\theta_3) \\ &+(1-\theta_1)\theta_2(1-\theta_3) + (1-\theta_1)(1-\theta_2)\theta_3] \\ &+\ln[1-\theta_1(1-\theta_3) - (1-\theta_1)\theta_3].\end{aligned}$$

Depending on a choice for initial parameter settings $\boldsymbol{\theta}^{(0)}$, the EM algorithm for symmetric causal independence models converges to one of the maxima:

$$\begin{aligned}&CLL(\boldsymbol{\theta})_{\max} \\ &= \begin{cases} 0 & \text{at } \boldsymbol{\theta} = (0,1,0), \\ -1.386 & \text{at } \boldsymbol{\theta} \in \{\left(\theta_1, 0, \frac{1}{2}\right), \left(\frac{1}{2}, 0, \theta_3\right)\}. \end{cases}\end{aligned}$$

Obviously, only the point $\boldsymbol{\theta} = (0,1,0)$ is a global maximum of the log-likelihood function while the other obtained points are local maxima.

The discussed counterexample proves that in general case the EM algorithm for symmetric causal independence models does not necessarily converge to the global maximum.

## 5 Experimental Results

For our experiments we use Reuters data collection, which allows us to evaluate the classification performance of large symmetric causal independence models where the number of cause variables for some document classes is in the hundreds.

### 5.1 Evaluation Scheme

Since we do not have an efficient algorithm to perform a search in the space of symmetric Boolean functions, we chose to model the interaction among cause and effect variables by means of Boolean threshold functions, which seem to be the most probable interaction functions for the given domains.

Given the model parameters $\boldsymbol{\theta}$, the testing data $\mathbf{D}_{test}$ and the classification threshold $\frac{1}{2}$, the classifications and misclassifications for both classes are computed. Let *tp (true positives)* stand for the number of data samples $(\mathbf{c}^j, e^{j+}) \in \mathbf{D}_{test}$ for which $\Pr(e^+ \mid \mathbf{c}^j, \boldsymbol{\theta}) \geq \frac{1}{2}$ and *fp (false*

*positives)* stand for the number of data samples $(\mathbf{c}^j, e^{j+}) \in \mathbf{D}_{test}$ for which $\Pr(e^+ \mid \mathbf{c}^j, \boldsymbol{\theta}) < \frac{1}{2}$. Likewise, *tn (true negatives)* is the number of data samples $(\mathbf{c}^j, e^{j-}) \in \mathbf{D}_{test}$ for which $\Pr(e^+ \mid \mathbf{c}^j, \boldsymbol{\theta}) < \frac{1}{2}$ and *fp (false positives)* is the number of data samples $(\mathbf{c}^j, e^{j-}) \in \mathbf{D}_{test}$ for which $\Pr(e^+ \mid \mathbf{c}^j, \boldsymbol{\theta}) \geq \frac{1}{2}$. To evaluate the classification performance we use *accuracy*, which is a measure of correctly classified cases,

$$\eta = \frac{tp + tn}{tp + tn + fn + fp},$$

and *F-measure*, which combines *precision* $\pi = \frac{tp}{tp+fp}$ and *recall* $\rho = \frac{tp}{tp+fn}$,

$$F = \frac{2\pi\rho}{\pi + \rho}.$$

### 5.2 Reuters Data Set

We used the Reuters-21578 text categorization collection containing the Reuters new stories preprocessed by Karčiauskas (2002). The training set contained 7769 documents and the testing set contained 3018 documents. For every of the ten document classes the most informative features were selected using the expected information gain as a feature selection criteria, and each document class was classified separately against all other classes. We chose to use the same threshold for the expected information gain as in (Vomlel, 2006), the number of selected features varied from 23 for the corn document class to 307 for the earn document class. While learning the values of the hidden parameters the EM algorithm was stopped after 50 iterations. The accuracy and F-measure for causal independence models with the threshold interaction function $k = 1, \ldots, 4$ are given in tables 1 and 2. Even though the threshold to select the relevant features was tuned for the noisy OR classifier, for 5 document classes the causal independence models with other interaction function than logical OR provided better results.

The accuracy and F-measure of the noisy OR model and a few other classifiers on the Reuters data collection reported in (Vomlel, 2006) show the competitive performance of the noisy OR model.

Table 1: Classification accuracy for symmetric causal independence models with the interaction function $\tau_k, k = 1, \ldots, 4$ for Reuters data set; $N_{Class}$ is number of documents in the corresponding class.

| Class | $N_{Class}$ | $\tau_1$ | $\tau_2$ | $\tau_3$ | $\tau_4$ |
|---|---|---|---|---|---|
| earn | 1087 | 96.3 | **97.2** | **97.2** | 96.8 |
| acq | 719 | 93.1 | **93.2** | **93.2** | 93.0 |
| crude | 189 | **98.1** | **98.1** | 97.6 | 97.7 |
| money-fx | 179 | 95.8 | 95.8 | 95.9 | **96.0** |
| grain | 149 | **99.2** | 99.0 | 98.2 | 97.9 |
| interest | 131 | 96.5 | **96.8** | 96.7 | 96.7 |
| trade | 117 | 96.6 | 97.0 | **97.3** | **97.3** |
| ship | 89 | **98.9** | 98.8 | 98.7 | 98.6 |
| wheat | 71 | **99.5** | 99.2 | 98.8 | 98.5 |
| corn | 56 | **99.7** | 99.4 | 99.1 | 98.8 |

## 6 Discussion

In this paper, we discussed the application of symmetric causal independence models for classification. We developed the EM algorithm to learn the parameters in symmetric causal independence models and studied its convergence. The reported experimental results indicate that it is unnecessary to restrict causal independence models to only two interaction functions, logical OR and logical AND. Competitive classification performance of symmetric causal independence models present them as a potentially useful additional tool to the set of classifiers.

The current study has only examined the problem of learning conditional probabilities of hidden variables. The problem of learning an optimal interaction function has not been addressed. Efficient search in symmetric Boolean function space is a possible direction for future research.

### Acknowledgments

This research, carried out in the TimeBayes project, was supported by the Netherlands Organization for Scientific Research (NWO) under project number FN4556. The authors are grateful to Gytis Karčiauskas for the preprocessed Reuters data. We would also like to thank Jiří

Table 2: Classification F-measure for symmetric causal independence models with the interaction function $\tau_k, k = 1, \ldots, 4$ for Reuters data set; $N_{Class}$ is number of documents in the corresponding class.

| Class | $N_{Class}$ | $\tau_1$ | $\tau_2$ | $\tau_3$ | $\tau_4$ |
|---|---|---|---|---|---|
| earn | 1087 | 95.0 | **96.1** | **96.1** | 95.6 |
| acq | 719 | **85.3** | 84.3 | 84.5 | 83.8 |
| crude | 189 | 84.5 | **85.7** | 80.7 | 81.0 |
| money-fx | 179 | 60.9 | 62.1 | 62.6 | **62.7** |
| grain | 149 | **92.7** | 89.9 | 80.7 | 77.2 |
| interest | 131 | 40.2 | **55.0** | 53.3 | 54.0 |
| trade | 117 | 51.0 | 61.2 | **63.7** | **63.7** |
| ship | 89 | **79.5** | 77.7 | 74.5 | 71.5 |
| wheat | 71 | **90.3** | 81.8 | 71.4 | 66.2 |
| corn | 56 | **91.8** | 83.6 | 72.5 | 61.5 |

Vomlel for sharing his code and insights.

## References

S. Boyd and L. Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

J. Darroch. 1964. On the distribution of the number of successes in independent trials. *The Annals of Mathematical Statistics*, 35:1317–1321.

A.P. Dempster, N.M. Laird and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

F.J. Díez. 1993. Parameter adjustment in Bayes networks. The generalized noisy OR-gate. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 99–105.

H.B. Enderton. 1972. *A Mathematical Introduction to Logic*. Academic Press, San Diego.

D. Heckerman and J.S. Breese. 1994. A new look at causal independence. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 286–292.

M. Henrion. 1989. Some practical issues in constructing belief networks. *Uncertainty in Artificial Intelligence*, 3:161–173.

R. Jurgelenaite, T. Heskes and P.J.F. Lucas. 2006. Noisy threshold functions for modelling causal independence in Bayesian networks. *Technical report ICIS–R06014*, Radboud University Nijmegen.

H.J. Kappen and J.P. Neijt. 2002. Promedas, a probabilistic decision support system for medical diagnosis. *Technical report*, SNN - UMCU.

G. Karčiauskas. 2002. Text Categorization using Hierarchical Bayesian Network Classifiers. *Master thesis*, Aalborg University.

L. Le Cam. 1960. An approximation theorem for the Poisson binomial distribution. *Pacific Journal of Mathematics*, 10:1181–1197.

P.J.F. Lucas. 2005. Bayesian network modelling through qualitative patterns. *Artificial Intelligence*, 163:233–263.

C. Meek and D. Heckerman. 1997. Structure and parameter learning for causal independence and causal interaction models. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 366–375.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffman Publishers.

M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann and G.F. Cooper. 1991. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base, I – the probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–255.

S. Srinivas. 1993. A generalization of the noisy-or model. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 208–215.

S. Visscher, P.J.F. Lucas, M. Bonten and K. Schurink. 2005. Improving the therapeutic performance of a medical Bayesian network using noisy threshold models. In *Proceedings of the Sixth International Symposium on Biological and Medical Data Analysis*, pages 161–172.

J. Vomlel. 2006. Noisy-or classifier. *International Journal of Intelligent Systems*, 21:381–398.

I. Wegener. 1987. *The Complexity of Boolean Functions*. John Wiley & Sons, New York.

N.L. Zhang and D. Poole. 1996. Exploiting causal independence in Bayesian networks inference. *Journal of Artificial Intelligence Research*, 5:301–328.

# Complexity Results for Enhanced Qualitative Probabilistic Networks

Johan Kwisthout and Gerard Tel
Department of Information and Computer Sciences
University of Utrecht
Utrecht, The Netherlands

**Abstract**

While quantitative probabilistic networks (QPNs) allow the expert to state influences between nodes in the network as influence signs, rather than conditional probabilities, inference in these networks often leads to ambiguous results due to unresolved trade-offs in the network. Various enhancements have been proposed that incorporate a notion of strength of the influence, such as enhanced and rich enhanced operators. Although inference in standard (i.e., not enhanced) QPNs can be done in time polynomial to the length of the input, the computational complexity of inference in such enhanced networks has not been determined yet. In this paper, we introduce relaxation schemes to relate these enhancements to the more general case where continuous influence intervals are used. We show that inference in networks with continuous influence intervals is *NP*-hard, and remains *NP*-hard when the intervals are discretised and the interval $[-1, 1]$ is divided into blocks with length of $\frac{1}{4}$. We discuss membership of *NP*, and show how these general complexity results may be used to determine the complexity of specific enhancements to QPNs. Furthermore, this might give more insight in the particular properties of feasible and infeasible approaches to enhance QPNs.

## 1 Introduction

While probabilistic networks (Pearl, 1988) are based on a intuitive notion of causality and uncertainty of knowledge, elicitating the required probabilistic information from the experts can be a difficult task. Qualitative probabilistic networks (Wellman, 1990), or QPNs, have been proposed as a qualitative abstraction of probabilistic networks to overcome this problem. These QPNs summarise the conditional probabilities between the variables in the network by a sign, which denotes the direction of the effect. In contrast to quantitative networks, where inference has been shown to be *NP*-hard (Cooper, 1990), these networks have efficient (i.e., polynomial-time) inference algorithms. QPNs are often used as an intermediate step in the construction of a probabilistic network (Renooij and van der Gaag, 2002), as a tool for verifying properties of such networks (van der Gaag et al., 2006), or in applications where the exact probability distribution is unknown or irrelevant (Wellman, 1990).

Nevertheless, this qualitative abstraction leads to ambiguity when influences with contrasting signs are combined. *Enhanced* QPNs have been proposed (Renooij and van der Gaag, 1999) in order to allow for more flexibility in determining the influences (e.g., *weakly* or *strongly* positive) and partially resolve conflicts when combining influences. Also, mixed networks (Renooij and van der Gaag, 2002) have been proposed, to facilitate stepwise quantification and allowing both qualitative and quantitative influences to be modelled in the network.

Although inference in quantitative networks is *NP*-hard, and polynomial-time algorithms are known for inference in standard qualitative networks, the computational complexity of inference in enhanced networks has not been deter-

mined yet. In this paper we recall the definition of QPNs in section 2, and we introduce a framework to relate various enhancements, such as enhanced, rich enhanced, and interval-based operators in section 3. In section 4 we show that inference in the general, interval-based case is *NP*-hard. In section 5 we show that it remains *NP*-hard if we use discrete - rather than continuous - intervals. Furthermore, we argue that, although hardness proofs might be nontrivial to obtain, it is unlikely that there exist polynomial algorithms for less general variants of enhanced networks, such as the enhanced and rich enhanced operators suggested by Renooij and Van der Gaag (1999). Finally, we conclude our paper in section 6.

## 2 Qualitative Probabilistic Networks

In qualitative probabilistic networks, a directed acyclic graph $G = (V, A)$ is associated with a set $\Delta$ of qualitative influences and synergies (Wellman, 1990), where the influence of one node to another is summarised by a sign[1]. For example, a *positive* influence of a node $A$ on its successor $B$, denoted with $S^+(A, B)$, expresses that higher values for $A$ make higher values for $B$ more likely than lower values, regardless of influences of other nodes on $B$. In binary cases, with $a > \bar{a}$ and $b > \bar{b}$, this can be summarised as $\Pr(b \mid ax) - \Pr(b \mid \bar{a}x) \geq 0$ for any value of $x$ of other predecessors of $B$. *Negative* influences, denoted by $S^-$, and *zero* influences, denoted by $S^0$, are defined analogously. If an influence is not positive, negative, or zero, it is *ambiguous*, denoted by $S^?$. Influences can be direct (causal influence) or *induced* (inter-causal influence or *product synergy*). In the latter case, the value of one node influences the probabilities of values of another node, given a third node (Druzdzel and Henrion, 1993b).

Various properties hold for these qualitative influences, namely *symmetry*, *transitivity*, *composition*, *associativity* and *distribution* (Wellman, 1990; Renooij and van der Gaag, 1999). If we define $\hat{S}^\delta(A, B, t_i)$ as the influence $S^\delta$, with $\delta \in \{+, -, 0, ?\}$, from a node $A$ on a node $B$ along trail $t_i$, we can formalise these properties as shown in table 1. The $\otimes$- and $\oplus$-operators that follow from the transitivity and composition properties are defined in table 2.

| | |
|---|---|
| symmetry | $\hat{S}^\delta(A, B, t_i) \in \Delta \Leftrightarrow \hat{S}^\delta(B, A, t_i^{-1}) \in \Delta$ |
| transitivity | $\hat{S}^\delta(A, B, t_i) \wedge \hat{S}^{\delta'}(B, C, t_j) \Rightarrow \hat{S}^{\delta \otimes \delta'}(A, C, t_i \circ t_j)$ |
| composition | $\hat{S}^\delta(A, B, t_i) \wedge S^{\delta'}(A, B, t_j) \Rightarrow S^{\delta \oplus \delta'}(A, B, t_i \circ t_j)$ |
| associativity | $S^{(\delta \oplus \delta') \oplus \delta''} = S^{\delta \oplus (\delta' \oplus \delta'')}$ |
| distribution | $S^{(\delta \oplus \delta') \otimes \delta''} = S^{(\delta \otimes \delta'') \oplus (\delta' \otimes \delta'')}$ |

Table 1: Properties of qualitative influences

| $\otimes$ | + | − | 0 | ? |
|---|---|---|---|---|
| + | + | − | 0 | ? |
| − | − | + | 0 | ? |
| 0 | 0 | 0 | 0 | 0 |
| ? | ? | ? | 0 | ? |

| $\oplus$ | + | − | 0 | ? |
|---|---|---|---|---|
| + | + | ? | + | ? |
| − | ? | − | − | ? |
| 0 | + | − | 0 | ? |
| ? | ? | ? | ? | ? |

Table 2: The $\otimes$- and $\oplus$-operator for combining signs

Using these properties, an efficient (polynomial time) inference algorithm can be constructed (Druzdzel and Henrion, 1993a) that propagates observed node values to other neighbouring nodes. The basic idea of the algorithm, given in pseudo-code in figure 1, is as follows. When entering the procedure, a node $I$ is instantiated with a '+' or a '−' (i.e., *trail* = $\emptyset$, *from* = *to* = $I$ and *msign* = '+' *or* '−'). Then, this node sign is propagated through the network, following active trails and updating nodes when needed. Observe from table 2 that a node can change at most two times: from '0' to '+', '−', or '?', and then only to '?'. This algorithm visits each node at most two times, and therefore halts after a polynomial amount of time.

[1]Note that the network $Q = (G, \Delta)$ represents infinitely many *quantitative* probabilistic networks that respect the restrictions on the conditional probabilities denoted by the signs of all arcs.

**procedure** PropagateSign(*trail*, *from*, *to*, *msign*):
sign[*to*] ← sign[*to*] ⊕ *msign*;
*trail* ← *trail* ∪ { *to* };
**for** each active neighbour $V_i$ of *to*
**do** *lsign* ← sign of influence between *to* and $V_i$;
*msign* ← sign[*to*] ⊗ *lsign*;
**if** $V_i \notin$ *trail* **and** sign[$V_i$] ≠ sign[$V_i$] ⊕ *msign*
**then** PropagateSign(*trail*, *to*, $V_i$, *msign*).

Figure 1: The sign-propagation algorithm

## 3 Enhanced QPNs

These qualitative influences and synergies can of course be extended to preserve a larger amount of information in the abstraction (Renooij and van der Gaag, 1999). For example, given a certain cut-off value $\alpha$, an influence can be *strongly* positive ($\Pr(b \mid ax) - \Pr(b \mid \bar{a}x) \geq \alpha$) or *weakly* negative ($-\alpha \leq \Pr(b \mid ax) - \Pr(b \mid \bar{a}x) \leq 0$). The basic '+' and '−' signs are enhanced with signs for strong influences ('++' and '−−') and augmented with multiplication indices to handle complex dependencies on $\alpha$ as a result of transitive and compositional combinations. In addition, signs such as '$+_?$' and '$-_?$' are used to denote positive or negative influences of unknown strength. Using this notion of strength, trade-offs in the network can be modelled by compositions of weak and strong opposite signs.

Furthermore, an *interval network* can be constructed (Renooij and van der Gaag, 2002), where each arc has an associated influence interval rather than a sign. Such an influence is denoted as $F^{[p,q]}(A, B)$, meaning that $\Pr(b \mid ax) - \Pr(b \mid \bar{a}x) \in [p, q]$. Note that, given this definition, $S^+(A, B) \iff F^{[0,1]}(A, B)$, and similar observations hold for $S^-$, $S^0$ and $S^?$. We will denote the intervals $[-1, 0]$, $[0, 1]$, $[0, 0]$ and $[-1, 1]$ as *unit intervals*, being special cases that correspond to the traditional qualitative networks. The ⊗- and ⊕-operator, denoting transitivity and composition in interval networks are defined in table 3. Note that it is possible that a result of a combination of two trails leads to an empty set, for example when combining $[\frac{1}{2}, 1]$ with $[\frac{3}{4}, 1]$, which would denote that the total influence of a node on another node, along multiple trails, would be greater than one, which is impossible. Since the individual intervals might be estimated by experts, this situation is not unthinkable, especially in large networks. This property can be used to detect design errors in the network.

Note, that the symmetry, associativity, and distribution property of qualitative networks do no longer apply in these enhancements. For example, although a positive influence from a node $A$ to $B$ along the direction of the arc also has a positive influence in the opposite direction, the strength of this influence is unknown. Also, the outcome of the combination of a strongly positive, weakly positive and weakly negative sign depends on the evaluation order.

| $\otimes_i$ | $[r, s]$ |
|---|---|
| $[p, q]$ | $[\min X, \max X]$, where $X = \{p \cdot r, p \cdot s, q \cdot r, q \cdot s\}$ |

| $\oplus_i$ | $[r, s]$ |
|---|---|
| $[p, q]$ | $[p + r, q + s] \cap [-1, 1]$ |

Table 3: The $\otimes_i$- and $\oplus_i$-operators for interval multiplication and addition

### 3.1 Relaxation schemes

If we take a closer look at the $\oplus_e$, $\oplus_r$, and $\otimes_e$ operators defined in (Renooij and van der Gaag, 1999) and compare them with the interval operators $\oplus_i$ and $\otimes_i$, we can see that the interval results are sometimes somehow 'relaxed'. We see that symbols representing influences correspond to intervals, but after the application of any operation on these intervals, the result is extended to an interval that can be represented by one of the available symbols. For example, in the interval model we have $[\alpha, 1] \oplus_i [-1, 1] = [\alpha - 1, 1]$, but, while $[\alpha, 1]$ corresponds to ++ in the enhanced model and $[-1, 1]$ corresponds to ?, $+ + \oplus_e ? = ? \equiv [-1, 1]$. The lower limit $\alpha - 1$ is relaxed to $-1$, because the actually resulting interval $[\alpha - 1, 1]$ does not correspond to any symbol. Therefore, to connect the (enhanced) qualitative and interval models, we will introduce *relaxation schemes* that map the result of each operation to the minimal interval that can be represented by

one of the available symbols:

**Definition 1.** (Relaxation scheme)
$R_x$ will be defined as a relaxation scheme, denoted as $R_x([a,b]) = [c,d]$, if $R_x$ maps the outcome $[a,b]$ of an $\oplus$ or $\otimes$ operation to an interval $[c,d]$, where $[a,b] \subseteq [c,d]$.

In standard QPNs, the relaxation scheme (which I will denote $R_I$ or the *unit scheme*) is defined as in figure 2.

$$R_I(a,b) = \begin{cases} [0,1] & \text{if } a \geq 0 \wedge b > 0 \\ [-1,0] & \text{if } a < 0 \wedge b \leq 0 \\ [0,0] & \text{if } a = b = 0 \\ [-1,1] & \text{otherwise} \end{cases}$$

Figure 2: Relaxation scheme $R_I(a,b)$

Similarly, the $\oplus_e$, $\oplus_r$, and $\otimes_e$ operators can be denoted with the relaxation schemes in figure 3, in which $m$ equals $\min(i,j)$ and $\alpha$ is an arbitrary cut-off value. To improve readability, in the remainder of this paper the $\oplus$- and $\otimes$-operators, when used without index, denote operators on intervals as defined in table 3.

$$R_{\otimes_e}(a,b) = \begin{cases} [-1,1] & \text{if } a < 0 \wedge b > 0 \\ [a,b] & \text{otherwise} \end{cases}$$

$$R_{\oplus_e}(a,b) = \begin{cases} [\alpha^m, 1] & \text{if } a = \alpha^i + \alpha^j \leq b \\ [-1, -\alpha^m] & \text{if } b = -(\alpha^i + \alpha^j) \geq a \\ [0,1] & \text{if } a \leq b = \alpha^i + \alpha^j \\ [-1,0] & \text{if } a = -(\alpha^i + \alpha^j) \leq b \\ [0,1] & \text{if } a = (\alpha^i - \alpha^j) \\ & \text{and } b \geq 0 \text{ and } i < j \\ [-1,0] & \text{if } a = -(\alpha^i - \alpha^j) \\ & \text{and } b \leq 0 \text{ and } i < j \\ [-1,1] & \text{if } a \leq 0 \text{ and } b \geq 0 \\ [a,b] & \text{otherwise} \end{cases}$$

$$R_{\oplus_r}(a,b) = \begin{cases} [-1,1] & \text{if } a < 0 \wedge b > 0 \\ [a,b] & \text{otherwise} \end{cases}$$

Figure 3: Relaxation schemes $R_{\otimes_e}$, $R_{\oplus_e}$, and $R_{\oplus_r}$

This notion of a relaxation scheme allows us to relate various operators in a uniform way. A common property of most of these schemes is that the $\oplus$-operator is no longer associative. The result of inference now depends on the order in which various influences are propagated through the network.

### 3.2 Problem definition

To decide on the complexity of inference of this general, interval-based enhancements of QPNs, a decision problem needs to be determined. We state this problem, denoted as IPIEQNETD[2], as follows.

IPIEQNETD
**Instance:** Qualitative Probabilistic Network $Q = (G, \Delta)$ with an instantiation for $A \in V(G)$ and a node $B \in V \setminus \{A\}$.
**Question:** Is there an ordering on the combination of influences such that the influence of $A$ on $B \subset [-1,1]$?

### 3.3 Probability representation

In this paper, we assume that the probabilities in the network are represented by fractions, denoted by integer pairs, rather than by reals. This has the advantage, that the length of the result of addition and multiplication of fractions is polynomial in the length of the original numbers. We can efficiently code the fractions in the network by rewriting them, using their least common denominator. Adding or multiplying these fractions will not affect their denominators, whose length will not change during the inference process.

## 4 Complexity of the problems

We will prove the hardness of the inference problem IPIEQNETD by a transformation from 3SAT. We construct a network $Q$ using clauses $C$ and boolean variables $U$, and prove that, upon instantiation of a node $I$ to $[1,1]$, there is an ordering on the combination of influences such that the influence of $I$ on a given node $Y \in Q \setminus \{I\}$ is a true subset of $[-1,1]$, if and only if the corresponding 3SAT instance is satisfiable. In the network, the influence of a node

[2] An acronym for *Interval-based Probabilistic Inference in Enhanced Qualitative Networks*

$A$ on a node $B$ along the arc $(A, B)$ is given as an interval; when the interval equals $[1, 1]$ (i.e., $\Pr(b \mid a) = 1$ and $\Pr(b \mid \bar{a}) = 0$) then the interval is omitted for readability. Note, that the influence of $B$ on $A$, against the direction of the arc $(A, B)$, equals the unit interval of the influence associated with $(A, B)$.

As a running example, we will construct a network for the following 3SAT instance, introduced in (Cooper, 1990):

**Example 1.** ($3\text{SAT}_{ex}$)
$U = \{u_1, u_2, u_3, u_4\}$, and $C = \{(u_1 \vee u_2 \vee u_3), (\neg u_1 \vee \neg u_2 \vee u_3), (u_2 \vee \neg u_3 \vee u_4)\}$

This instance is satisfiable, for example with the truth assignment $u_1 = T$, $u_2 = F$, $u_3 = F$, and $u_4 = T$.

### 4.1 Construct for our proofs

For each variable in the 3SAT instance, our network contains a "variable gadget" as shown in figure 4. After the instantiation of node $I$ with $[1, 1]$, the influence at node $D$ equals $[\frac{1}{2}, 1] \oplus [-\frac{1}{2}, \frac{1}{2}] \oplus [-1, -\frac{1}{2}]$, which is either $[-1, \frac{1}{2}]$, $[-\frac{1}{2}, 1]$ or $[-1, 1]$, depending on the order of evaluation. We will use the non-associativity of the $\oplus$-operator in this network as a non-deterministic choice of assignment of truth values to variables. As we will see later, an evaluation order that leads to $[-1, 1]$ can be safely dismissed (it will act as a 'falsum' in the clauses, making both $x$ and $\neg x$ false), so we will concentrate on $[-\frac{1}{2}, 1]$ (which will be our $T$ assignment) and $[-1, \frac{1}{2}]$ ($F$ assignment) as the two possible choices.

We construct literals $u_i$ from our 3SAT instance, each with a variable gadget $Vg$ as input. Therefore, each variable can have a value of $[-1, \frac{1}{2}]$ or $[-\frac{1}{2}, 1]$ as influence, non-deterministicly. Furthermore, we add clause-networks $Cl_j$ for each clause in the instance and connect each variable $u_i$ to a clause-network $Cl_j$ if the variable occurs in $C_j$. The influence associated with this arc $(u_i, Cl_j)$ is defined as $F^{[p,q]}(u_i, Cl_j)$, where $[p, q]$ equals $[-1, 0]$ if $\neg u_i$ is in $C_j$, and $[0, 1]$ if $u_i$ is in $C_j$ (figure 5). Note

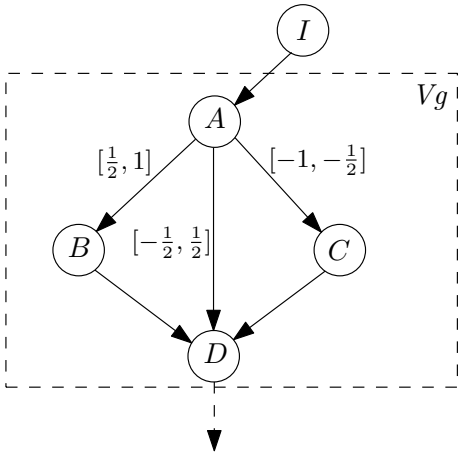



Figure 4: "Variable gadget" $Vg$

that an $\otimes$-operation with $[-1, 0]$ will transform a value of $[-1, \frac{1}{2}]$ in $[-\frac{1}{2}, 1]$ and vice versa, and $[0, 1]$ will not change them. We can therefore regard an influence $F^{[-1,0]}$ as a negation of the truth assignment for that influence. Note, that $[-1, 1]$ will stay the same in both cases.

If we zoom in on the clause-network in figure 6, we will see that the three 'incoming' variables in a clause, that have a value of either $[-1, \frac{1}{2}]$, $[-\frac{1}{2}, 1]$, or $[-1, 1]$, are multiplied with the arc influence $F_{i,j}$ to form variables, and then combined with the instantiation node (with a value of $[1, 1]$), forming nodes $w_i$. Note that $[-1, \frac{1}{2}] \oplus [1, 1] = [-1, 1] \oplus [1, 1] = [0, 1]$ and that $[-\frac{1}{2}, 1] \oplus [1, 1] = [\frac{1}{2}, 1]$. Since a $[-1, 1]$ result in the variable gadget does not change by multiplying with the $F_{i,j}$ influence, and leads to the same value as an $F$ variable, such an assignment will never satisfy the 3SAT instance.

The influences associated with these nodes $w_i$ are multiplied by $[\frac{1}{2}, 1]$ and added together in the clause result $O_j$. At this point, $O_j$ has a value of $[\frac{k}{4}, 1]$, where $k$ equals the number of literals which are true in this clause. The consecutive adding to $[-\frac{1}{4}, 1]$, multiplication with $[0, 1]$ and adding to $[1, 1]$ has the function of a *logical or*-operator, giving a value for $C_j$ of $[\frac{3}{4}, 1]$ if no literal in the clause was true, and $[1, 1]$ if one or more were true.

We then combine the separate clauses $C_j$ into a variable $Y$, by adding edges from each clause to $Y$ using intermediate variables $D_1$ to $D_{n-1}$.

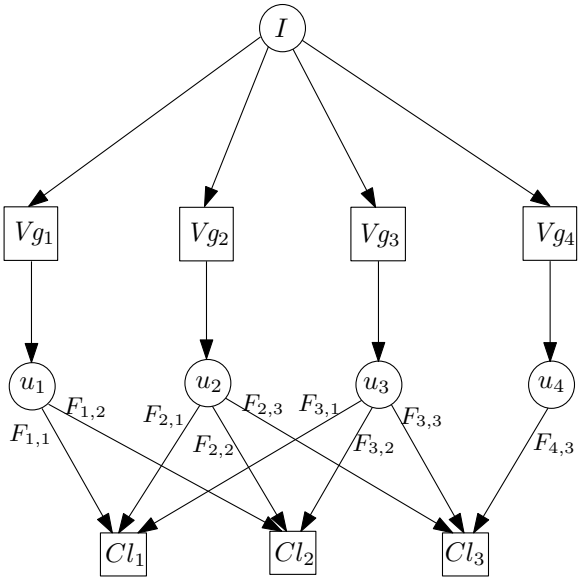



Figure 5: The literal-clause construction

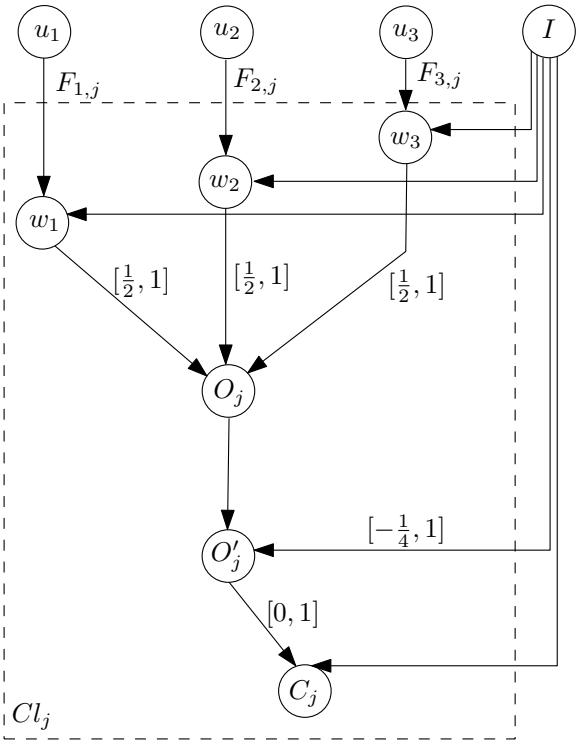



Figure 6: The clause construction

The use of these intermediate variables allows us to generalise these results to more restricted cases. The interval of these edges is $[\frac{1}{2},1]$, leading to a value of $[1,1]$ in $Y$ if and only if all clauses $C_j$ have a value of $[1,1]$ (see figure 7). The influence interval in $Y$ has a value between $[\frac{3}{4},1]$ and $[\frac{2^{k+1}-1}{2^{k+1}},1]$, where $k$ is the number of clauses, if one or more clauses had a value of $[0,1]$. Note that we can easily transform the interval in $Y$ to a subset of $[-1,1]$ in $Y'$ by consecutively adding it to $[-1,1]$ and $[-1+\frac{1}{2^{k+1}},1]$. This would result in a true subset of $[-1,1]$ if and only $Y$ was equal to $[1,1]$.

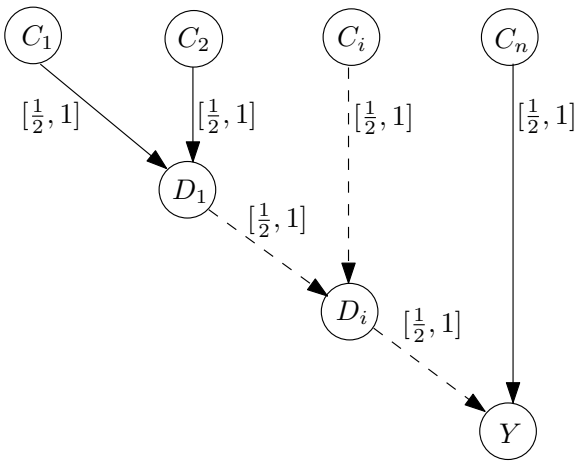



Figure 7: Connecting the clauses

### 4.2 *NP*-hardness proof

Using the construct presented in the previous section, the computational complexity of the iPieqnetd can be established as follows.

**Theorem 1.** *The* iPieqnetd *problem is NP-hard.*

*Proof.* To prove *NP*-hardness, we construct a transformation from the 3sat problem. Let $(U,C)$ be an instance of this problem, and let $Q_{(U,C)}$ be the interval-based qualitative probabilistic network constructed from this instance, as described in the previous section. When the node $I \in Q$ is instantiated with $[1,1]$, then $I$ has an influence of $[1,1]$ on $Y$ (and therefore an influence on $Y'$ which is a true subset of $[-1,1]$) if and only if all nodes $C_j$ have a value of $[1,1]$, i.e. there exists an ordering of the operators in the 'variable-gadget' such that at least one literal in $C$ is true. We conclude that $(U,C)$ has a solution with at least one true literal in each clause, if and only if the iPieqnetd problem has a solution for network $Q_{(U,C)}$, instanti-

ation $I = [1, 1]$ and node $Y'$. Since $Q_{(U,C)}$ can be computed from $(U, C)$ in polynomial time, we have a polynomial-time transformation from 3SAT to iPIEQNETD, which proves *NP*-hardness of iPIEQNETD. □

### 4.3 On the possible membership of *NP*

Although iPIEQNETD has been shown to be *NP*-hard, membership of the *NP*-class (and, as a consequence, *NP*-completeness) is not trivial to prove. To prove membership of *NP*, one has to prove that if the instance is solvable, then there exists a certificate, polynomial in length with respect to the input of the problem, that can be used to verify this claim. A trivial certificate could be a formula, using the $\oplus$- and $\otimes$-operators, influences, and parentheses, describing how the influence of the a certain node can be calculated from the instantiated node and the characteristics of the network. Unfortunately, such a certificate can grow exponentially large.

In special cases, this certificate might also be described by an ordering of the nodes in the network and for each node an ordering of the inputs, which would be a polynomial certificate. For example, the variable gadget from figure 4 can be described with the ordering $I, A, B, C, D$ plus an ordering on the incoming trails in $D$. All possible outcomes of the propagation algorithm can be calculated using this description. Note, however, that there are other propagation sequences possible that cannot be described in this way. For example, the algorithm might first explore the trail $A \rightarrow B \rightarrow D \rightarrow C$, and after that the trail $A \rightarrow C \rightarrow D \rightarrow B$. Then, the influence in $D$ is dependent on the information in $C$, but $C$ is visited by $D$ following the first trail, and $D$ is visited by $C$ if the second trail is explored. Nevertheless, this cannot lead to other outcomes than $[-1, \frac{1}{2}]$, $[-\frac{1}{2}, 1]$, or $[-1, 1]$.

However, it is doubtful that such a description exists for all possible networks. For example, even if we can make a topological sort of a certain network, starting with a certain instantiation node $X$ and ending with another node $Y$, it is still not the case that a propagation sequence that follows only the direction of the arcs until all incoming trails of $Y$ have been calculated always yields better results than a propagation sequence that doesn't have this property. This makes it plausible that there exist networks where an optimal solution (i.e., a propagation sequence that leads to the smallest possible subset of $[-1, 1]$ at the node we are interested in) cannot be described using such a polynomial certificate.

## 5 Operator variants

In order to be able to represent every possible 3SAT instance, a relaxation scheme must be able to generate a variable gadget, and retain enough information to discriminate between the cases where zero, one, two or three literals in each clause are true. Furthermore, the relaxation scheme must be able to represent the instantiations $[1, 1]$ (or $\top$) and $[-1, -1]$ (or $\bot$), and the uninstantiated case $[0, 0]$. With a relaxation scheme that effectively divides the interval $[-1, 1]$ in discrete blocks with size of a multitude of $\frac{1}{4}$, (such as $R_{\frac{1}{4}}(a, b) = [\frac{\lfloor 4a \rfloor}{4}, \frac{\lceil 4b \rceil}{4}]$) the proof construction is essentially the same as in the general case discussed in section 3. This relaxation scheme does not have any effect on the intervals we used in the variable gadget and the clause construction of $Q_{(U,C)}$, the network constructed in the *NP*-hardness proof of the general case used only intervals $(a, b)$ for which $R_{\frac{1}{4}}(a, b) = (a, b)$. Furthermore, when connecting the clauses, the possible influences in $Y$ are relaxed to $[0, 1]$, $[\frac{1}{4}, 1]$, $[\frac{1}{2}, 1]$, $[\frac{3}{4}, 1]$, and $[1, 1]$, so we can onstruct $Y'$ by consecutively adding the interval in $Y$ to $[-1, 1]$ and $[-\frac{3}{4}, 1]$. Thus, the problem - which we will denote as RELAXED-PIEQNETD - remains *NP*-hard for relaxation scheme $R_{\frac{1}{4}}$.

The non-associativity of the $\oplus_e$- and $\oplus_r$-operators defined in (Renooij and van der Gaag, 1999) suggest hardness of the inference problem as well. Although $\oplus_e$ and $\oplus_r$ are not associative, they cannot produce results that can be regarded as opposites. For example, the expression $(++\oplus_e+\oplus_e-)$ can lead to a positive influence of unknown strength ('$+_?$') when evaluated as $((++\oplus_e+)\oplus_e-)$ or an unknown influence

('?') when evaluated as $(+ + \oplus_e (+ \oplus_e -))$, but never to a negative influence. A transformation from a 3SAT variant might not succeed because of this reason. However, it might be possible to construct a transformation from RELAXED-PIEQNETD, which is subject of ongoing research.

## 6 Conclusion

In this paper, we addressed the computational complexity of inference in enhanced Qualitative Probabilistic Networks. As a first step, we have "embedded" both standard and enhanced QPNs in the interval-model using relaxation schemes, and we showed that inference in this general interval-model is *NP*-hard and remains *NP*-hard for relaxation scheme $R_{\frac{1}{4}}(a, b)$. We believe, that the hardness of inference is due to the fact that reasoning in QPNs is underdefined: The outcome of the inference process depends on choices during evaluation. Nevertheless, further research needs to be conducted in order to determine where exactly the *NP/P* border lies, in other words: which enhancements to the standard qualitative model allow for polynomial-time inference, and which enhancements lead to intractable results. Furthermore, a definition of transitive and compositional combinations of qualitative influences in which the outcome is independent of the order of the influence propagation might reduce the computational complexity of inference and facilitate the use of qualitative models to design, validate, analyse, and simulate probabilistic networks.

### Acknowledgements

This research has been (partly) supported by the Netherlands Organisation for Scientific Research (NWO).
The authors wish to thank Hans Bodlaender and Silja Renooij for their insightful comments on this subject.

## References

G.F. Cooper. 1990. The Computational Complexity of Probabilistic Inference using Bayesian Belief Networks. *Artificial Intelligence*, 42(2):393–405.

M.J. Druzdzel and M. Henrion. 1993a. Belief Propagation in Qualitative Probabilistic Networks. In N. Piera Carrete and M.G. Singh, editors, *Qualitative Reasoning and Decision Technologies*, CIMNE: Barcelona.

M.J. Druzdzel and M. Henrion. 1993b. Intercausal reasoning with uninstantiated ancestor nodes. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 317–325.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, Palo Alto.

S. Renooij and L.C. van der Gaag. 1999. Enhancing QPNs for trade-off resolution. In *Proceedings of the Fifteenth Conference on Uncertainy in Artificial Intelligence*, pages 559–566.

S. Renooij and L.C. van der Gaag. 2002. From Qualitative to Quantitative Probabilistic Networks. In *Proceedings of the Eighteenth Conference in Uncertainty in Artificial Intelligence*, pages 422–429.

L.C. van der Gaag, S. Renooij, and P.L. Geenen. 2006. Lattices for verifying monotonicity of Bayesian networks. In *Proceedings of the Third European Workshop on Probabilistic Graphical Models (to appear)*.

M.P. Wellman. 1990. Fundamental Concepts of Qualitative Probabilistic Networks. *Artificial Intelligence*, 44(3):257–303.

# Decision analysis with influence diagrams using Elvira's explanation facilities

Manuel Luque and Francisco J. Díez
Dept. Inteligencia Artificial, UNED
Juan del Rosal, 16, 28040 Madrid, Spain

**Abstract**

Explanation of reasoning in expert systems is necessary for debugging the knowledge base, for facilitating their acceptance by human users, and for using them as tutoring systems. Influence diagrams have proved to be effective tools for building decision-support systems, but explanation of their reasoning is difficult, because inference in probabilistic graphical models seems to have little relation with human thinking. The current paper describes some explanation capabilities for influence diagrams and how they have been implemented in Elvira, a public software tool.

## 1 Introduction

Influence diagrams (IDs) are a probabilistic graphical model for modelling decision problems. They constitute a simple graphical formalism that makes it possible to represent the three main components of decision problems: the uncertainty, due to the existence of variables not controlled by the decision maker, the decisions or actions under their direct control, and their preferences about the possible outcomes.

In the context of expert systems, either probabilistic or heuristic, the development of explanation facilities is important for three main reasons (Lacave and Díez, 2002). First, because the construction of those systems with the help of human experts is a difficult and time-consuming task, prone to errors and omissions. An explanation tool can help the experts and the knowledge engineers to *debug* the system when it does not yield the expected results and even before a malfunction occurs. Second, because human beings are reluctant to *accept* the advice offered by a machine if they are not able to understand how the system arrived at those recommendations; this reluctancy is especially clear in medicine (Wallis and Shortliffe, 1984). And third, because an expert system used as an intelligent *tutor* must be able to communicate to the apprentice the knowledge it contains, the way in which the knowledge has been applied for arriving at a conclusion, and what would have happened if the user had introduced different pieces of evidence (what-if reasoning).

These reasons are especially relevant in the case of probabilistic expert systems, because the elicitation of probabilities is a difficult task that usually requires debugging and refinement, and because the algorithms for the computation of probabilities and utilities are, at least apparently, very different from human reasoning.

Unfortunately, most expert systems and commercial tools available today, either heuristic or probabilistic, have no explanation capabilities. In this paper we describe some explanation methods developed as a response to the needs that we have detected when building and debugging medical expert systems (Díez et al., 1997; Luque et al., 2005) and when teaching probabilistic graphical models to pre- and postgraduate students of computer science and medicine (Díez, 2004). These new methods have been implemented in Elvira, a public software tool.

### 1.1 Elvira

Elvira[1] is a tool for building and evaluating graphical probabilistic models (Elvira Consor-

[1] At http://www.ia.uned.es/∼elvira it is possible to obtain the source code and several technical documents about Elvira.

tium, 2002) developed as a join project of several Spanish universities. It contains a graphical interface for editing networks, with specific options for canonical models, exact and approximate algorithms for both discrete and continuous variables, explanation facilities, learning methods for building networks from databases, etc. Although some of the algorithms can work with both discrete and continuous variables, most of the explanation capabilities assume that all the variables are discrete.

## 2 Influence diagrams

### 2.1 Definition of an ID

An influence diagram (ID) consists of a directed acyclic graph that contains three kinds of nodes: *chance nodes* $\mathbf{V}_C$, *decision nodes* $\mathbf{V}_D$ and *utility nodes* $\mathbf{V}_U$—see Figure 1. Chance nodes represent random variables not controlled by the decision maker. Decision nodes correspond to actions under the direct control of the decision maker. Utility nodes represent the decision maker's preferences. Utility nodes can not be parents of chance or decision nodes. Given that each node represents a variable, we will use the terms variable and node interchangeably.

In the extended framework proposed by Tatman and Shachter (1990) there are two kinds of utility nodes: *ordinary utility nodes*, whose parents are decision and/or chance nodes, and *super-value nodes*, whose parents are utility nodes, and can be in turn of two types, sum and product. We assume that there is a utility node $U_0$, which is either the only utility node or a descendant of all the other utility nodes, and therefore has no children.[2]

There are three kinds of arcs in an ID, depending on the type of node they go into. Arcs into chance nodes represent probabilistic dependency. Arcs into decision nodes represent availability of information, i.e., an arc $X \rightarrow D$ means that the state of $X$ is known when making decision $D$. Arcs into utility nodes represent functional dependence: for ordinary utility nodes, they represent the domain of the associated utility function; for a super-value node they indicate that the associated utility is a function (sum or product) of the utility functions of its parents.

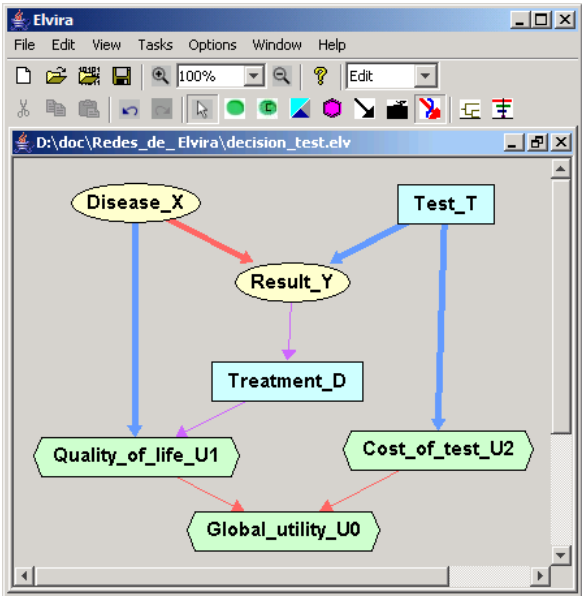



Figure 1: ID with two decisions (ovals), two chance nodes (squares) and three utility nodes (diamonds). There is a directed path including all the decisions and node $U_0$.

Standard IDs require that there is a directed path that includes all the decision nodes and indicates the order in which the decisions are made. This in turn induces a partition of $\mathbf{V}_C$ such that for an ID having $n$ decisions $\{D_0, \ldots, D_{n-1}\}$, the partition contains $n+1$ subsets $\{\mathbf{C}_0, \mathbf{C}_1, ..., \mathbf{C}_n\}$, where $\mathbf{C}_i$ is the set of chance variables $C$ such that there is a link $C \rightarrow D_i$ and no link $C \rightarrow D_j$ with $j < i$; i.e., $\mathbf{C}_i$ represents the set of random variables known for $D_i$ and unknown for previous decisions. $\mathbf{C}_n$ is the set of variables having no link to any decision, i.e., the variables whose true value is never known directly.

Given a chance or decision variable $V$, two decisions $D_i$ and $D_j$ such that $i < j$, and two links $V \rightarrow D_i$ and $V \rightarrow D_j$, the former link

[2] An ID that does not fulfill this condition can be transformed by adding a super-value node $U_0$ of type sum whose parents are the utility nodes that did not have descendants. The expected utility and the optimal strategy (both defined below) of the transformed diagram are the same as those of the original one.

is said to be a *no-forgetting link*. In the above example, $T \rightarrow D$ would be a non-forgetting link.

The variables known to the decision maker when deciding on $D_i$ are called *informational predecessors* of $D_i$ and denoted by $IPred(D_i)$. Assuming the *no-forgetting hypothesis*, we have that $IPred(D_i) = IPred(D_{i-1}) \cup \{D_{i-1}\} \cup \mathbf{C}_i = \mathbf{C}_0 \cup \{D_0\} \cup \mathbf{C}_1 \cup \ldots \cup \{D_{i-1}\} \cup \mathbf{C}_i$.

The quantitative information that defines an ID is given by assigning to each random node $C$ a probability distribution $P(c|pa(C))$ for each configuration of its parents, $pa(C)$, assigning to each ordinary utility node $U$ a function $\psi_U(pa(U))$ that maps each configuration of its parents onto a real number, and assigning a utility-combination function to each super-value node. The domain of each function $U$ is given by its *functional predecessors*, $FPred(U)$. For an ordinary utility node, $FPred(U) = Pa(U)$, and for a super-value node $FPred(U) = \bigcup_{U' \in Pa(U)} FPred(U')$.

For instance, in the ID of Figure 1, we have that $FPred(U_1) = \{X, D\}$, $FPred(U_2) = \{T\}$ and $FPred(U_0) = \{X, D, T\}$.

In order to simplify our notation, we will sometimes assume without loss of generality that for any utility node $U$ we have that $FPred(U) = \mathbf{V}_C \cup \mathbf{V}_D$.

### 2.2 Policies and expected utilities

For each configuration $\mathbf{v}_D$ of the decision variables $\mathbf{V}_D$ we have a joint distribution over the set of random variables $\mathbf{V}_C$:

$$P(\mathbf{v}_C : \mathbf{v}_D) = \prod_{C \in \mathbf{V}_C} P(c|pa(C)) \qquad (1)$$

which represents the probability of configuration $\mathbf{v}_C$ when the decision variables are externally set to the values given by $\mathbf{v}_D$ (Cowell et al., 1999).

A *stochastic policy* for a decision $D$ is a probability distribution defined over $D$ and conditioned on the set of its informational predecessors, $P_D(d|IPred(D))$. If $P_D$ is degenerate (consisting of ones and zeros only) then we say the policy is deterministic.

A *strategy* $\Delta$ for an ID is a set of policies, one for each decision, $\{P_D|D \in \mathbf{V}_D\}$. A strategy $\Delta$ induces a joint distribution over $\mathbf{V}_C \cup \mathbf{V}_D$ defined by

$$\begin{aligned} &P_\Delta(\mathbf{v}_C, \mathbf{v}_D) \\ &\quad = P(\mathbf{v}_C : \mathbf{v}_D) \prod_{D \in \mathbf{V}_D} P_D(d|IPred(D)) \\ &\quad = \prod_{C \in \mathbf{V}_C} P(c|pa(C)) \prod_{D \in \mathbf{V}_D} P_D(d|pa(D)) \end{aligned} \qquad (2)$$

Let $I$ be an ID, $\Delta$ a strategy for $I$ and $\mathbf{r}$ a configuration defined over a set of variables $\mathbf{R} \subseteq \mathbf{V}_C \cup \mathbf{V}_D$ such that $P_\Delta(\mathbf{r}) \neq 0$. The probability distribution *induced by strategy* $\Delta$ *given the configuration* $\mathbf{r}$, defined over $\mathbf{R}' = (\mathbf{V}_C \cup \mathbf{V}_D) \setminus \mathbf{R}$, is given by:

$$P_\Delta(\mathbf{r}'|\mathbf{r}) = \frac{P_\Delta(\mathbf{r}, \mathbf{r}')}{P_\Delta(\mathbf{r})} \; . \qquad (3)$$

Using this distribution we can compute the *expected utility of* $U$ *under strategy* $\Delta$ *given the configuration* $\mathbf{r}$ as:

$$EU_U(\Delta, \mathbf{r}) = \sum_{\mathbf{r}'} P_\Delta(\mathbf{r}'|\mathbf{r}) \psi_U(\mathbf{r}, \mathbf{r}') \; . \qquad (4)$$

For the terminal utility node $U_0$, $EU_{U_0}(\Delta, \mathbf{r})$ is said to be the *expected utility of strategy* $\Delta$ *given the configuration* $\mathbf{r}$, and denoted by $EU(\Delta, \mathbf{r})$.

We define the *expected utility of* $U$ *under strategy* $\Delta$ as $EU_U(\Delta) = EU_U(\Delta, \blacklozenge)$, where $\blacklozenge$ is the empty configuration. We also define the *expected utility of strategy* $\Delta$ as $EU(\Delta) = EU_{U_0}(\Delta)$. We have that

$$EU_U(\Delta) = \sum_{\mathbf{r}} P_\Delta(\mathbf{r}) EU_U(\Delta, \mathbf{r}) \; . \qquad (5)$$

An *optimal strategy* is a strategy $\Delta_{opt}$ that maximizes the expected utility:

$$\Delta_{opt} = \arg\max_{\Delta \in \Delta^*} EU(\Delta) \; , \qquad (6)$$

where $\Delta^*$ is the set of all strategies for $I$. Each policy in an optimal strategy is said to be an *optimal policy*. The *maximum expected utility* (MEU) is

$$MEU = EU(\Delta_{opt}) = \max_{\Delta \in \Delta^*} EU(\Delta) \; . \qquad (7)$$

The evaluation of an ID consists in finding the *MEU* and an optimal strategy. It can be proved (Cowell et al., 1999; Jensen, 2001) that

$$MEU = \sum_{\mathbf{c}_0} \max_{d_0} \ldots \sum_{\mathbf{c}_{n-1}} \max_{d_{n-1}} \sum_{\mathbf{c}_n} P(\mathbf{v}_C : \mathbf{v}_D)\psi(\mathbf{v}_C, \mathbf{v}_D) \; . \quad (8)$$

For instance, the *MEU* for the ID in Figure 1, assuming that $U_0$ is of type sum, is

$$MEU = \max_t \sum_y \max_d \sum_x P(x) \cdot P(y|t,x) \cdot \cdot \underbrace{(U_1(x,d) + U_2(t))}_{U_0(x,d,t)} \; . \quad (9)$$

### 2.3 Cooper policy networks

When a strategy $\Delta = \{P_D | D \in \mathbf{V}_D\}$ is defined for an ID, we can convert this into a Bayesian network, called *Cooper policy network* (CPN), as follows: each decision $D$ is replaced by a chance node with probability potential $P_D$ and parents $IPred(D)$, and each utility node $U$ is converted into a chance node whose parents are its functional predecessors (cf. Sec. 2.1); the values of new chance variables are $\{+u, \neg u\}$ and its probability is $P_{CPN}(+u|FPred(U)) = norm_U(U(FPred(U)))$, where $norm_U$ is a bijective linear transformation that maps the utilities onto the interval $[0, 1]$ (Cooper, 1988).

For instance, the CPN for the ID in Figure 1 is displayed in Figure 2. Please note the addition of the non-forgetting link $T \rightarrow D$ and that the parents of node $U_0$ are no longer $U_1$ and $U_2$ but $T$, $X$, and $D$, which were chance or decision nodes in the ID.

The joint distribution of the CPN is:

$$P_{CPN}(\mathbf{v}_C, \mathbf{v}_D, \mathbf{v}_U) = P_\Delta(\mathbf{v}_C, \mathbf{v}_D) \prod_{U \in \mathbf{V}_U} P_U(u|pa(U)) \; . \quad (10)$$

For a configuration $\mathbf{r}$ defined over a set of variables $\mathbf{R} \subseteq \mathbf{V}_C \cup \mathbf{V}_D$ and $U$ a utility node, it is possible to prove that

$$P_{CPN}(\mathbf{r}) = P_\Delta(\mathbf{r}) \quad (11)$$

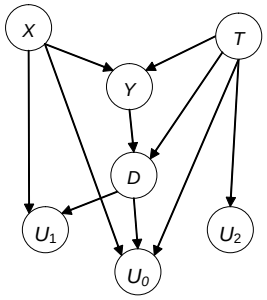



Figure 2: Cooper policy network (PN) for the ID in Figure 1.

$$P_{CPN}(+u|\mathbf{r}) = norm_U(EU_U(\Delta, \mathbf{r})) \; . \quad (12)$$

This equation allows us to compute $EU_U(\Delta, \mathbf{r})$ as $norm_U^{-1}(P_{CPN}(+u|\mathbf{r}))$; i.e., the expected utility for a node $U$ in the ID can be computed from the marginal probability of the corresponding node in the CPN.

## 3 Explanation of influence diagrams in Elvira

### 3.1 Explanation of the model

The explanation of IDs in Elvira is based, to a great extent, on the methods developed for explanation of Bayesian networks (Lacave et al., 2000; Lacave et al., 2006b). One of the methods that have proven to be more useful is the automatic colorings of links. The definitions in (Lacave et al., 2006b) for the sign of influence and magnitude of influence, inspired on (Wellman, 1990), have been adapted to incoming arcs to ordinary utility nodes.

Specifically, the *magnitude of the influence* gives a relative measure of how a variable is influencing an ordinary utility node (see (Lacave et al., 2006a) for further details). Then, the influence of a link pointing to a utility node is positive when higher values of $A$ lead to higher utilities. Cooper's transformation $norm_U$ guarantees that the magnitude of the influence is normalized.

For instance, in Figure 1 the link $X \rightarrow Y$ is colored in red because it represents a positive influence: the presence of the disease increases the probability of a positive result of the test. The link $X \rightarrow U_1$ is colored in blue because it

represents a negative influence: the disease decreases the expected quality of life. The link $D \rightarrow U_1$ is colored in purple because the influence that it represents is undefined: the treatment is beneficial for patients suffering from $X$ but detrimental for healthy patients.

Additionally, when a decision node has been assigned a policy, either by optimization or imposed by the user (see Sec. 3.3), the corresponding probability distribution $P_D$ can be used to color the links pointing to that node, as shown in Figure 1.

The coloring of links has been very useful in Elvira for debugging Bayesian networks (Lacave et al., 2006b) and IDs, in particular for detecting some cases in which the numerical probabilities did not correspond to the qualitative relations determined by human experts, and also for checking the effect that the informational predecessors of a decision node have on its policy.

### 3.2 Explanation of reasoning: cases of evidence

In Section 2.3 we have seen that, given a strategy, an ID can be converted into a CPN, which is a true Bayesian network. Consequently, all the explanation capabilities for BNs, such as Elvira's ability to manage evidence cases are also available for IDs.

A *finding* states with certainty the value taken on a chance or decision variable. A set of findings is called *evidence* and corresponds to a certain configuration $\mathbf{e}$ of a set of observed variables $\mathbf{E}$. An *evidence case* is determined by an evidence $\mathbf{e}$ and the posterior probabilities and the expected utilities induced by it.

A distinguishable feature of Elvira is its ability to manage several evidence cases simultaneously. A special evidence case is the *prior case*, which is the first case created and corresponds to the absence of evidence.

One of the evidence cases is marked as the *current case*. Its probabilities and utilities are displayed in the nodes. For the rest of the cases, the probabilities and utilities are displayed only by bars, in order to visually compare how they vary when new evidence is introduced.

The information displayed for nodes depends on the kind of node. Chance and decision nodes present bars and numbers corresponding to posterior probabilities of their states, $P_\Delta(v|\mathbf{e})$, as given by Equation 3. This is the probability that a chance variable takes a certain value or the probability that the decision maker chooses a certain option (Nilsson and Jensen, 1998)—please note that in Equation 3 there is no distinction between chance and decision nodes. Utility nodes show the expected utilities, $EU_U(\Delta, \mathbf{e})$, given by Equation 4. The guide bar indicates the range of the utilities.

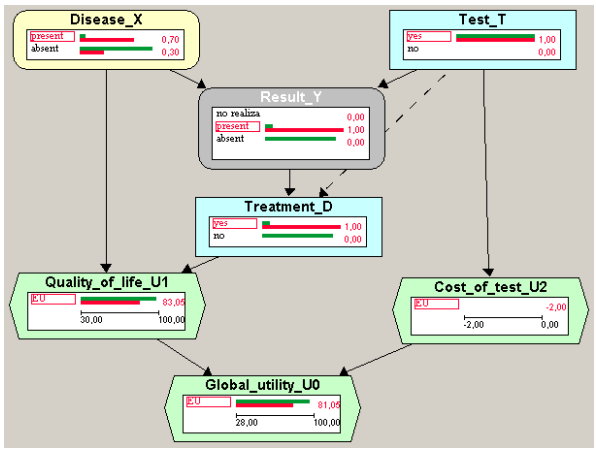



Figure 3: ID in Elvira with two evidence cases: (a) the prior case (no evidence); (b) a case with the evidence $\mathbf{e} = \{+y\}$.

Figure 3 shows the result of evaluating the ID in Figure 1. The link $T \rightarrow D$ is drawn as a discontinuous arrow to indicate that it has been added by the evaluation of the diagram. In this example, as there was no policy imposed by the user (see below), Elvira computed the optimal strategy. In this figure two evidence cases are displayed. The first one is the *prior case*, i.e., the case in which there is no evidence. The second evidence case is given by $\mathbf{e} = \{+y\}$; i.e., it displays the probabilities and utilities of the subpopulation of patients in which the test gives a positive result. Node $Y$ is colored in gray to highlight the fact that there is evidence about it. The probability of $+x$, represented by a red bar, is 0.70; the green bar close to it represents the probability of $+x$ for

the prior case, i.e., the prevalence of the disease; the red bar is longer than the green one because $P(+x|+y) > P(+x)$. The global utility for the second case is 81.05, which is smaller than the green bar close to it (the expected utility for the general population) because the presence of the symptom worsens the prognosis. The red bar for Treatment=yes is the probability that a patient having a positive result in the test receives the treatment; this probability is 100% because the optimal strategy determines that all symptomatic patients must be treated.

The possibility of introducing evidence in Elvira has been useful for building IDs in medicine: when we were interested in computing the posterior probability of diagnoses given several sets of findings, we need to manually convert the ID into a Bayesian network by removing decision and utility nodes, and each time the ID was modified we have to convert it into a Bayesian network to compute the probabilities. Now the probabilities can be computed directly on the ID.

#### 3.2.1 Clarifying the concept of evidence in influence diagrams

In order to avoid confusions, we must mention that Ezawa's method (1998) for introducing evidence in IDs is very different from the way that is introduced in Elvira. For Ezawa, the introduction of evidence $\mathbf{e}$ leads to a different decision problem in which the values of the variables in $\mathbf{E}$ would be known with certainty before making any decision. For instance, introducing evidence $\{+x\}$ in the ID in Figure 1 would imply that $X$ would be known when making decisions $T$ and $D$. Therefore, the expected utility of the new decision problem would be

$$\max_t \sum_y \max_d P(y|+x:t,d) \cdot \underbrace{(U_1(+x,d) + U_2(t))}_{U_0(+x,d,t)} .$$

where $P(y|+x:t,d) = P(+x,y:t,d)/P(+x) = P(y|+x:t)$. In spite of the apparent similarity of this expression with Equation 9, the optimal strategy changes significantly to "always treat, without testing", because if we know with certainty that the disease $X$ is present the result of the test is irrelevant. The $MEU$ for this decision problem is $U_1(+x,+d)$.

In contrast, the introduction of evidence in Elvira (which may include "findings" for decision variables as well) does not lead to a new decision scenario nor to a different strategy, since the strategy is determined *before* introducing the "evidence". Put another way, in Elvira we adopt the point view of an external observer of a system that includes the decision maker as one of its components. The probabilities and expected utilities given by Equations 2 and 4 are those corresponding to the subpopulation indicated by $\mathbf{e}$ when treated with strategy $\Delta$. For instance, given the evidence $\{+x\}$, the probability $P_\Delta(+t|+x)$ shown by Elvira is the probability that a patient suffering from $X$ receives the test, which is 100% (it was 0% in Ezawa's scenario), and $P_\Delta(+d|+x)$ is the probability that he receives the treatment; contrary to Ezawa's scenario, this probability may differ from 100% because of false negatives. The expected utility for a patient suffering from $X$ is

$$EU(\Delta,\{+x\}) = \\ = \sum_{t,y,d} P_\Delta(t,y,d|+x) \cdot \underbrace{(U_1(+x,d) + U_2(t))}_{U_0(+x,d,t)} .$$

where $P_\Delta(t,y,d|+x) = P_\Delta(t) \cdot P(y|t,+x) \cdot P_\Delta(d|t,y)$.

Finally, we must underlie that both approaches are not rivals. They correspond to different points of view when considering evidence in IDs and can complement each other in order to perform a better decision analysis and to explain the reasoning.[3]

### 3.3 What-if reasoning: analysis of non-optimal strategies

In Elvira it is possible to have a strategy in which some of the policies are imposed by the user and the others are computed by maximization. The way of imposing a policy consists in setting a probability distribution $P_D$ for the corresponding decision $D$ by means of Elvira's

[3] In the future, we will implement in Elvira Ezawa's method and the possibility of computing the expected value of perfect information (EVPI).

GUI; the process is identical to editing the conditional probability table of a chance node. In fact, such a decision will be treated by Elvira as it were a chance node, and the maximization is performed only on the rest of the decision nodes.

This way, in addition to computing the optimal strategy (when the user has imposed no policy), as any other software tool for influence diagrams, Elvira also permits to analyze how the expected utilities and the rest of the policies would vary if the decision maker chose a non-optimal policy for some of the decisions (what-if reasoning).

The reason for implementing this explanation facility is that when we were building a certain medical influence diagram (Luque et al., 2005) our expert wondered why the model recommended not to perform a certain test. We wished to compute the a posteriori probability of the disease given a positive result in the test, but we could not introduce this "evidence", because it was incompatible with the optimal policy (not to test). After we implemented the possibility of imposing non-optimal policies (in this case, performing the test) we could see that the posterior probability of the disease remained below the treatment threshold even after a positive result in the test, and given that the result of the test would be irrelevant, it is not worthy to do it.

### 3.4 Decision trees

Elvira can expand a decision tree corresponding to an ID, with the possibility of expanding and contracting its branches to the desired level of detail. This is especially useful when building IDs in medicine, because physicians are used to thinking in terms of scenarios and most of them are familiar with decision trees, while very few have heard about influence diagrams. One difference of Elvira with respect to other software tools is the possibility of expanding the nodes in the decision tree to explore the decomposition of its utility given by the structure of super-value nodes of the ID.

### 3.5 Sensitivity analysis

Recently Elvira has been endowed with some well-known sensitivity analysis tools, such as one-way sensitivity analysis, tornado diagrams, and spider diagrams, which can be combined with the above-mentioned methods for the explanation of reasoning. For instance, given the ID in Figure 1, one-way sensitivity analysis on the prevalence of the disease $X$ can be used to determine the threshold treatment, and we can later see whether the result of test $Y$ makes the probability of $X$ cross that threshold. In the construction of more complex IDs this has been useful for understanding why some tests are necessary or not, and why sometimes the result of a test is irrelevant.

## 4 Conclusions

The explanation of reasoning in expert systems is necessary for debugging the knowledge base, for facilitating their acceptance by human users, and for using them as tutoring systems. This is especially important in the case of influence diagrams, because inference in probabilistic graphical models seems to have little relation with human thinking. Nevertheless, in general current software tools for building and evaluating decision trees and IDs offer very little support for analyzing the "reasoning": usually they only permit to compute the value of information, to perform some kind of sensitivity analysis, or to expand a decision tree, which can be hardly considered as explanation capabilities.

In this paper we have described some facilities implemented in Elvira that have been useful for understanding the knowledge contained in a certain ID, and why its evaluation has led to certain results, i.e., the optimal strategy and the expected utility. They can analyze how the posterior probabilities, policies, and expected utilities would vary if the decision maker applied a different (non-optimal) strategy. Most of these explanation facilities are based on the construction of a so-called Cooper policy network, which is a true Bayesian network, and consequently all the explanation options that were implemented for Bayesian networks in Elvira are also avail-

able for IDs, such as the possibility of handling several evidence cases simultaneously. Another novelty of Elvira with respect to most software tools for IDs is the ability to include super-value nodes in the IDs, even when expanding the decision tree equivalent to an ID.

We have also mentioned in this paper how these explanation facilities, which we have developed as a response to the needs that we have encountered when building medical models, have helped us to explain the IDs to the physicians collaborating with us and to debug those models (see (Lacave et al., 2006a) for further details).

**Acknowledgments**

Part of this work has been financed by the Spanish CICYT under project TIC2001-2973-C05. Manuel Luque has also been partially supported by Department of Education of the Comunidad de Madrid and the European Social Fund (ESF).

## References

G. F. Cooper. 1988. A method for using belief networks as influence diagrams. In *Proceedings of the 4th Workshop on Uncertainty in AI*, pages 55–63, University of Minnesota, Minneapolis.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

F. J. Díez, J. Mira, E. Iturralde, and S. Zubillaga. 1997. DIAVAL, a Bayesian expert system for echocardiography. *Artificial Intelligence in Medicine*, 10:59–73.

F.J. Díez. 2004. Teaching probabilistic medical reasoning with the Elvira software. In R. Haux and C. Kulikowski, editors, *IMIA Yearbook of Medical Informatics*, pages 175–180. Schattauer, Sttutgart.

The Elvira Consortium. 2002. Elvira: An environment for creating and using probabilistic graphical models. In *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM'02)*, pages 1–11, Cuenca, Spain.

K. Ezawa. 1998. Evidence propagation and value of evidence on influence diagrams. *Operations Research*, 46:73–83.

F. V. Jensen. 2001. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York.

C. Lacave and F. J. Díez. 2002. A review of explanation methods for Bayesian networks. *Knowledge Engineering Review*, 17:107–127.

C. Lacave, R. Atienza, and F. J. Díez. 2000. Graphical explanation in Bayesian networks. In *Proceedings of the International Symposium on Medical Data Analysis (ISMDA-2000)*, pages 122–129, Frankfurt, Germany. Springer-Verlag, Heidelberg.

C. Lacave, M. Luque, and F. J. Díez. 2006a. Explanation of Bayesian networks and influence diagrams in Elvira. Technical Report, CISIAD, UNED, Madrid.

C. Lacave, A. Oniśko, and F. J. Díez. 2006b. Use of Elvira's explanation facility for debugging probabilistic expert systems. *Knowledge-Based Systems*. In press.

M. Luque, F. J. Díez, and C. Disdier. 2005. Influence diagrams for medical decision problems: Some limitations and proposed solutions. In J. H. Holmes and N. Peek, editors, *Proceedings of the Intelligent Data Analysis in Medicine and Pharmacology*, pages 85–86.

D. Nilsson and F. V. Jensen. 1998. Probabilities of future decisions. In *Proceedings from the International Conference on Informational Processing and Management of Uncertainty in knowledge-based Systems*, pages 1454–1461.

J. A. Tatman and R. D. Shachter. 1990. Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 20:365–379.

J. W. Wallis and E. H. Shortliffe. 1984. Customized explanations using causal knowledge. In B. G. Buchanan and E. H. Shortliffe, editors, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, chapter 20, pages 371–388. Addison-Wesley, Reading, MA.

M. P. Wellman. 1990. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44:257–303.

# Dynamic importance sampling in Bayesian networks using factorisation of probability trees

Irene Martínez
Department of Languages and Computation
University of Almería
Almería, 04120, Spain

Carmelo Rodríguez and Antonio Salmerón
Department of Statistics and Applied Mathematics
University of Almería
Almería, 04120, Spain

**Abstract**

Factorisation of probability trees is a useful tool for inference in Bayesian networks. Probabilistic potentials some of whose parts are proportional can be decomposed as a product of smaller trees. Some algorithms, like lazy propagation, can take advantage of this fact. Also, the factorisation can be used as a tool for approximating inference, if the decomposition is carried out even if the proportionality is not completely reached. In this paper we propose the use of approximate factorisation as a means of controlling the approximation level in a dynamic importance sampling algorithm.

## 1 Introduction

In this paper we propose an algorithm for updating probabilities in Bayesian networks. This problem is known to be NP-hard even in the approximate case (Dagum and Luby, 1993). There exist several deterministic approximate algorithms (Cano et al., 2000; Cano et al., 2002; Cano et al., 2003; Kjærulff, 1994) as well as algorithms based on Monte Carlo simulation. The two main approaches are: Gibbs sampling (Jensen et al., 1995; Pearl, 1987) and importance sampling (Cheng and Druzdzel, 2000; Hernández et al., 1998; Moral and Salmerón, 2005; Salmerón et al., 2000; Shachter and Peot, 1990).

A class of these simulation procedures is composed by the importance sampling algorithms based on approximate pre-computation (Hernández et al., 1998; Moral and Salmerón, 2005; Salmerón et al., 2000). These methods perform first a fast but non exact propagation, consisting of a node removal process (Zhang and Poole, 1996). In this way, an approximate 'a posteriori' distribution is obtained. In the second stage a sample is drawn using the approximate distribution and the probabilities are estimated according to the importance sampling methodology (Rubinstein, 1981). In the most recent algorithm (Moral and Salmerón, 2005) a dynamic approach is adopted, in which the sampling distributions are updated according to the information collected during the simulation.

Recently, a new approach to construct approximate deterministic algorithms was proposed in (Martínez et al., 2005), based on the concept of approximate factorisation of the probability trees used to represent the probability functions.

The goal of this paper is to incorporate the ideas contained in (Martínez et al., 2005) to the dynamic algorithm introduced in (Moral and Salmerón, 2005), with the aim of showing that the approximation level can be controlled by the factorisation alternatively to pruning the trees.

The rest of the paper is organised as follows: in section 2 we analyse the main features of the dynamic importance sampling algorithm

in (Moral and Salmerón, 2005). Section 3 explain the concept of approximate factorisation of probability trees. Then, in section 4 we explain how both ideas can be combined in a new algorithm. The performance of the new algorithm is tested using three real-world Bayesian networks in section 5 and the paper ends with the conclusions in section 6.

## 2 Dynamic importance sampling in Bayesian networks

Along this paper we will consider a Bayesian network with variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ where each $X_i$ is a discrete variable taking values on a finite set $\Omega_{X_i}$. By $\Omega_{\mathbf{X}}$ we denote the state space of the $n$-dimensional random variable $\mathbf{X}$.

Probabilistic reasoning in Bayesian networks requires the computation of the posterior probabilities $p(x_k|\mathbf{e})$, $x_k \in \Omega_{X_k}$ for each variable of interest $X_k$, given that some other variables $\mathbf{E}$ have been observed to take value $\mathbf{e}$.

The posterior probability mentioned above can be expressed as

$$p(x_k|\mathbf{e}) = \frac{p(x_k, \mathbf{e})}{p(\mathbf{e})} \quad \forall x_k \in \Omega_{X_k} \ ,$$

and, since $p(\mathbf{e})$ is a constant value, the problem of calculating the posterior probability of interest is equivalent to obtaining $p(x_k, \mathbf{e})$ and normalising afterwards. If we denote by $p(\mathbf{x})$ the joint distribution for variables $\mathbf{X}$, then it holds that

$$p(x_k, \mathbf{e}) = \sum_{\Omega_{\mathbf{X}\setminus(\{X_k\}\cup\mathbf{E})}} p(\mathbf{x}) \ ,$$

where we assume that the $k$-th coordinate of $\mathbf{x}$ is equal to $x_k$ and the coordinates in $\mathbf{x}$ corresponding to observed variables are equal to $\mathbf{e}$. Therefore, the problem of probabilistic reasoning can be reduced to computing a sum, and here is where the *importance sampling* technique takes part.

Importance sampling is a well known method for computing integrals (Rubinstein, 1981) or sums over multivariate functions. A straightforward way to do that could be to draw a sample from $p(\mathbf{x})$ and then estimate $p(x_k, \mathbf{e})$ from it. However, $p(\mathbf{x})$ is often unmanageable, due to the large size of the state space of $\mathbf{X}$, $\Omega_{\mathbf{X}}$.

Importance sampling tries to overcome this problem by using a simplified probability function $p^*(\mathbf{x})$ to obtain a sample of $\Omega_{\mathbf{X}}$. The estimation is carried out according to the next procedure.

**Importance Sampling**

1. FOR $j := 1$ to $m$ (sample size)
   (a) Generate a configuration $\mathbf{x}^{(j)} \in \Omega_{\mathbf{X}}$ using $p^*$.
   (b) Compute a weight for the generated configuration as:
   $$w_j := \frac{p(\mathbf{x}^{(j)})}{p^*(\mathbf{x}^{(j)})} \ . \qquad (1)$$
2. For each $x_k \in \Omega_{X_k}$, estimate $p(x_k, \mathbf{e})$ as $\hat{p}(x_k, \mathbf{e})$ obtained as the sum of the weights in formula (1) corresponding to configurations containing $x_k$ divided by $m$.
3. Normalise the values $\hat{p}(x_k, \mathbf{e})$ in order to obtain $\hat{p}(x_k|\mathbf{e})$, i.e., an estimation of $p(x_k|\mathbf{e})$.

In (Salmerón et al., 2000; Moral and Salmerón, 2005), a sampling distribution is computed for each variable, so that $p^*$ is equal to the product of the sampling distributions for all the variables. The sampling distribution for each variable can be obtained through a process of variable elimination. Assume that the variables are eliminated following the order $X_1, \ldots, X_n$, and that, before eliminating the first variable, $H$ is the set of conditional distributions in the network. Then, the next algorithm obtains a sampling distribution for he $i$-th variable.

**Get Sampling Distribution($X_i$,$H$)**

1. $H_i := \{f \in H | f \text{ is defined for } X_i\}$.
2. $f_i := \prod_{f \in H_i} f$.
3. $f_i' := \sum_{x \in \Omega_{X_i}} f_i$.

4. $H := H \setminus H_i \cup \{f_i'\}$.

5. RETURN $(f_i)$.

Simulation is carried out in an order contrary to the one in which variables are deleted. Each variable $X_i$ is simulated from its sampling distribution $f_i$. This function is defined for variable $X_i$ and other variables already sampled. The potential $f_i$ is restricted to the already obtained values of the variables for which it is defined, except $X_i$, giving rise to a function which depends only on $X_i$. Finally, a value for this variable is obtained with probability proportional to the values of this potential. If all the computations are exact, it was proved in (Hernández et al., 1998) that, following this procedure, we are really sampling with the optimal probability $p^*(\mathbf{x}) = p(\mathbf{x}|\mathbf{e})$. However, the result of the combinations in the process of obtaining the sampling distributions may require a large amount of space to be stored, and therefore approximations are usually employed, either using probability tables (Hernández et al., 1998) or probability trees (Salmerón et al., 2000) to represent the distributions.

In (Moral and Salmerón, 2005) an alternative procedure to simulate each variable was used. Instead of restricting $f_i$ to the values of the variables already sampled, all the functions in $H_i$ are restricted, resulting in a set of functions depending only on $X_i$. The sampling distribution is then computed by multiplying all these functions. If the computations are exact, then both distributions are the same, as restriction and combination commute. This is the basis for the dynamic updating procedure proposed in (Moral and Salmerón, 2005). Probabilistic potentials are represented by probability trees, which are approximated by pruning some of its branches when computing $f_i$ during the calculation of the sampling distributions. This approximation is somehow corrected according to the information collected during the simulation: the probability value of the configuration already simulated provided by the product of the potentials in $H_i$ must be the same as the probability value for the same configuration provided by $f_i'$. Otherwise, potential $f_i$ is re-computed in order to correct the detected discrepancy.

## 3 Factorisation of probability trees

As mentioned in section 2, the approximation of the probability trees used to represent the probabilistic potentials consists of pruning some of their branches, namely those that lead to similar leaves (similar probability values), that can be substituted by the average, so that the error of the approximation depends on the differences between the values of the leaves corresponding to the pruned branches. Another way of taking advantage of the use of probability trees is given by the possible presence of proportionality between different subtrees (Martínez et al., 2002). This fact is illustrated in figures 1 and 2. In this case, the tree in figure 1 can be represented, without loss of precision, by the product of two smaller trees, shown in figure 2.

Note that the second tree in figure 2 does not contain variable $X$. It means that, when computing the sampling distribution for variable $X$, the products necessary to obtain function $f_i$ would be done over smaller trees.

### 3.1 Approximate Factorisation of Probability Trees

Factorisation of probability trees can be utilised as a tool for developing approximate inference algorithms, when the proportionality condition is relaxed. In this case, probability trees can be decomposed even if we find only *almost* proportional, rather than proportional, subtrees. Approximate factorisation and its application to Lazy propagation was proposed in (Martínez et al., 2005), and is stated as follows. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two sub-trees which are siblings for a given context (i.e. both sub-trees are children of the same node), such that both have the same size and their leaves contain only strictly positive numbers. The goal of the *approximate factorisation* is to find a tree $\mathcal{T}_2^*$ with the same structure than $\mathcal{T}_2$, such that $\mathcal{T}_2^*$ and $\mathcal{T}_1$ become proportional, under the restriction that the potential represented by $\mathcal{T}_2^*$ must be as close as possible to the one represented by $\mathcal{T}_2$. Then, $\mathcal{T}_2$ can be replaced by $\mathcal{T}_2^*$ and the resulting tree that

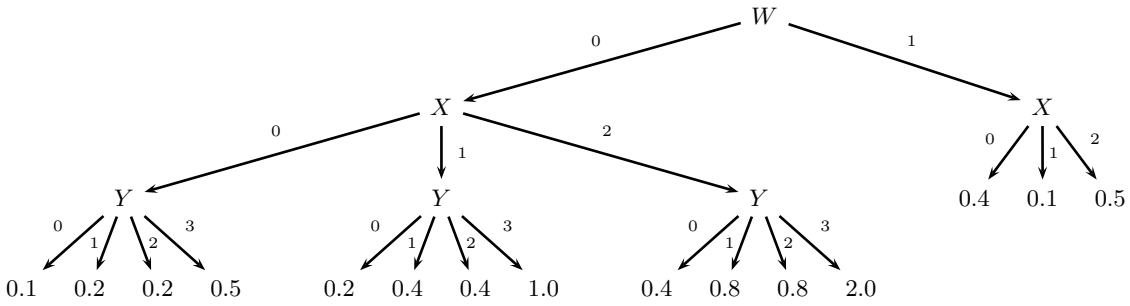



Figure 1: A probability tree proportional below $X$ for context $(W = 0)$.

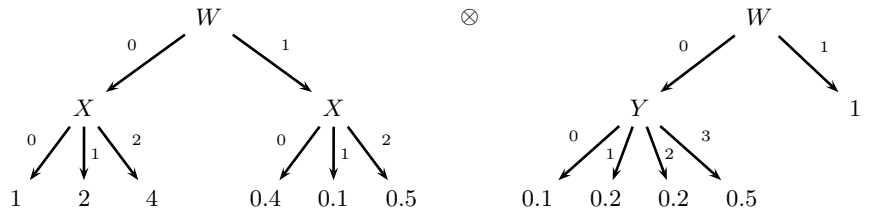



Figure 2: Decomposition of the tree in figure 1 with respect to variable $X$.

contain $\mathcal{T}_1$ and $\mathcal{T}_2$ can be decomposed by factorisation. Formally, approximate factorisation is defined through the concept of $\delta$-factorisability.

**Definition 1.** A probability tree $\mathcal{T}$ is *$\delta$-factorisable* within context $(\mathbf{X}_C = \mathbf{x}_C)$, with proportionality factors $\boldsymbol{\alpha}$ with respect to a divergence measure $\mathcal{D}$ if there is an $x_i \in \Omega_X$ and a set $L \subset \Omega_X \setminus \{x_i\}$ such that for every $x_j \in L$, $\exists \alpha_j > 0$ such that

$$\mathcal{D}(\mathcal{T}^{R(\mathbf{X}_C=\mathbf{x}_C, X=x_j)}, \alpha_j \cdot \mathcal{T}^{R(\mathbf{X}_C=\mathbf{x}_C, X=x_i)}) \leq \delta \ ,$$

where $\mathcal{T}^{R(\mathbf{X}_C=\mathbf{x}_C, X=x_i)}$ denotes the subtree of $\mathcal{T}$ which is reached by the branch where variables $\mathbf{X}_C$ take values $\mathbf{x}_C$ and $X_i$ takes value $x_i$. Parameter $\delta > 0$ is called the *tolerance of the approximation*.

Observe that if $\delta = 0$, we have exact factorisation. In the definition above, $\mathcal{D}$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$ are related in such a way that $\boldsymbol{\alpha}$ can be computed in order to minimise the value of the divergence measure $\mathcal{D}$.

In (Martínez et al., 2005) several divergence measures are considered, giving the optimal $\boldsymbol{\alpha}$ in each case. In this work we will consider only the measure that showed the best performance in (Martínez et al., 2005) that we will describe now. Consider a probability tree $\mathcal{T}$. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be sub-trees of $\mathcal{T}$ below a variable $X$, for a given context $(\mathbf{X}_C = \mathbf{x}_c)$ with leaves $\mathcal{P} = \{p_i : i = 1, \ldots, n \ ; \ p_i \neq 0\}$ and $\mathcal{Q} = \{q_i : i = 1, \ldots, n\}$ respectively. As we described before, approximate factorisation is achieved by replacing $\mathcal{T}_2$ by another tree $\mathcal{T}_2^*$ such that $\mathcal{T}_2^*$ is proportional to $\mathcal{T}_1$. It means that the leaves of $\mathcal{T}_2^*$ will be $\mathcal{Q}^* = \{\alpha p_i : i = 1, \ldots, n\}$, where $\alpha$ is the proportionality factor between $\mathcal{T}_1$ and $\mathcal{T}_2$. Let us denote by $\{\pi_i = q_i/p_i, i = 1, \ldots, n\}$ the ratios between the leaves of $\mathcal{T}_2$ and $\mathcal{T}_1$.

The $\chi^2$ divergence, defined as

$$\mathcal{D}_\chi(\mathcal{T}_2, \mathcal{T}_2^*) = \sum_{i=1}^{n} \frac{(q_i - \alpha p_i)^2}{q_i} \ ,$$

is minimised for $\alpha$ equal to

$$\alpha_\chi = \frac{\sum_{i=1}^n p_i}{\sum_{i=1}^n p_i/\pi_i} \ .$$

In this work we will use its normalised version

$$\mathcal{D}_{\chi^*}(\mathcal{T}_2, \mathcal{T}_2^*) = \sqrt{\frac{\mathcal{D}_\chi}{\mathcal{D}_\chi + n}} \ , \qquad (2)$$

which takes values between 0 and 1 and is minimised for the same $\alpha$.

## 4 Dynamic importance sampling combined with factorisation

As we mentioned in section 2, the complexity of the dynamic importance sampling algorithm relies on the computation of the sampling distribution. In the algorithm proposed in (Moral and Salmerón, 2005), this complexity is controlled by pruning the trees resulting from multiplications of other trees.

Here we propose to use approximate factorisation instead of tree pruning to control the complexity of the sampling distributions computation. Note that these two alternatives (approximate factorisation and tree pruning) are not exclusive. They can be used in a combined form. However, in order to have a clear idea of how approximate factorisation affects the accuracy of the results, we will not mix it with tree pruning in the algorithm proposed here.

The difference between the dynamic algorithm described in (Moral and Salmerón, 2005) and the method we propose in this paper is in the computation of the sampling distributions. The simulation phase and the update of the sampling distribution is carried out exactly in the same way, except that we have established a limit for the number of updates during the simulations equal to 1000 iterations. It means that, after iteration #1000 in the simulation, the sampling distributions are no longer updated. We have decided this with the aim of avoiding a high increase in space requirements during the simulation. Therefore, we only describe the part of the algorithm devoted to obtain the sampling distribution for a given variable $X_i$.

**Get Sampling Distribution($X_i$,$H$,$\mathcal{D}$,$\delta$)**

1. $H_i := \{f \in H | f \text{ is defined for } X_i\}$.
2. FOR each $f \in H_i$,
   (a) IF there is a context $\mathbf{X}_C$ for which the tree corresponding to $f$ is $\delta$-factorisable with respect to $\mathcal{D}$,
   - Decompose $f$ as $f_1 \times f_2$, where $f_1$ is defined for $X_i$ and $f_2$ is not.
   - $H_i := (H_i \setminus \{f\}) \cup \{f_1\}$.
   - $H := (H \setminus \{f\}) \cup \{f_2\}$.

   (b) ELSE
   - $H := H \setminus \{f\}$.
3. $f_i := \prod_{f \in H_i} f$.
4. $f_i' := \sum_{x \in \Omega_{X_i}} f_i$.
5. $H := H \cup \{f_i'\}$.
6. RETURN $(f_i)$.

According to the algorithm above, the sampling distribution for a variable $X_i$ is computed by taking all the functions which are defined for it, but unlike the method in (Moral and Salmerón, 2005), the product of all those functions is postponed until the possible factorisations of all the intervening functions are tested. Therefore, the product in step 3 is carried out over functions with smaller domains than the product in step 2 of the algorithm described in section 2.

The accuracy of the sampling distribution is controlled by parameter $\delta$, so that higher values of it result in worse approximations.

## 5 Experimental evaluation

In this section we describe a set of experiments carried out to show how approximate factorisation can be used to control the level of approximation in dynamic importance sampling. We have implemented the algorithm in java, as a part of the Elvira system (Elvira Consortium, 2002). We have selected three real-world Bayesian networks borrowed from the Machine Intelligence group at Aalborg University (`www.cs.aau.dk/research/MI/`). The

three networks are called Link (Jensen et al., 1995), Munin1 and Munin2 (Andreassen et al., 1989). Table 1 displays, for each network, the number of variables, number of observed variables (selected at random) and its *size*. By the size we mean the sum of the clique sizes that resulted when using HUGIN (Jensen et al., 1990) for computing the exact posterior distributions given the observed variables.

Table 1: Statistics about the networks used in the experiments.

| | Vars. | Obs. vars. | Size |
|---|---|---|---|
| Link | 441 | 166 | 23,983,962 |
| Munin1 | 189 | 8 | 83,735,758 |
| Munin2 | 1003 | 15 | 2,049,942 |

We have run the importance sampling algorithm with tolerance values $\delta = 0.1, 0.05, 0.01, 0.005$ and $0.001$ and with different number of simulation iterations: 2000, 3000, 4000 and 5000. Given the random nature of this algorithm, we have run each experiment 20 times and the errors have been averaged. For one variable $X_l$, the error in the estimation in its posterior distribution is measured as (Fertig and Mann, 1980):

$$G(X_l) = \sqrt{\frac{1}{|\Omega_{X_l}|} \sum_{a_l \in \Omega_{X_l}} \frac{(p'(a_l|\mathbf{e}) - p(a_l|\mathbf{e}))^2}{p(a_l|\mathbf{e})(1 - p(a_l|\mathbf{e}))}} \tag{3}$$

where $p(a_l|\mathbf{e})$ is the true *a posteriori* probability, $p'(a_l|\mathbf{e})$ is the estimated value and $|\Omega_{X_l}|$ is the number of cases of variable $X_l$. For a set of variables $\{X_1, \ldots, X_n\}$, the error is:

$$G(\{X_1, \ldots, X_n\}) = \sqrt{\sum_{i=1}^{n} G(X_i)^2} \tag{4}$$

This error measure emphasises the differences in small probability values, which means that is more discriminant than other measures like the mean squared error.

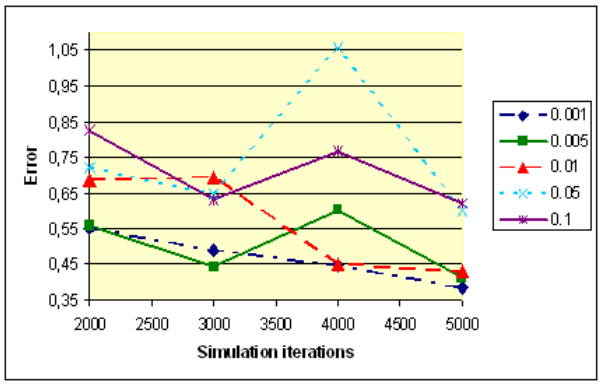



Figure 3: Results for Link with $\delta = 0.1, 0.05, 0.01, 0.005$ and $0.001$.

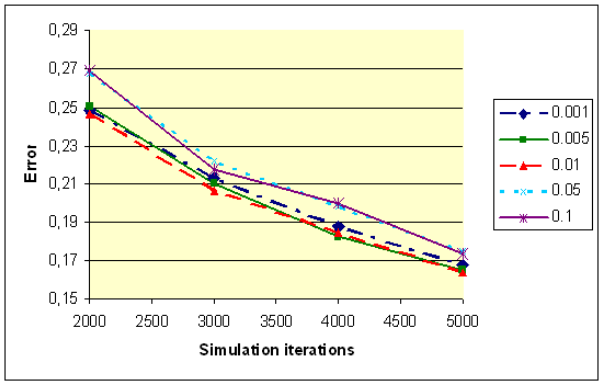



Figure 4: Results for Munin1 with $\delta = 0.1, 0.05, 0.01, 0.005$ and $0.001$.

### 5.1 Results discussion

The results summarised in figures 3, 4 and 5, indicate that the error can be somehow controlled by means of the $\delta$ parameter. There are, however, irregularities in the graphs, probably due to the variance associated with the estimations, due to the stochastic nature of the importance sampling method.

The best estimations are achieved for the Munin1 network. This is due to the fact that few factorisations are actually carried out, and therefore the number of approximations is lower, what also decreases the variance of the estimations.

Factorisation or even approximate factorisation are difficult to carry out over very extreme distributions. In the approximate case, a high value for $\delta$ would be necessary. This difficulty

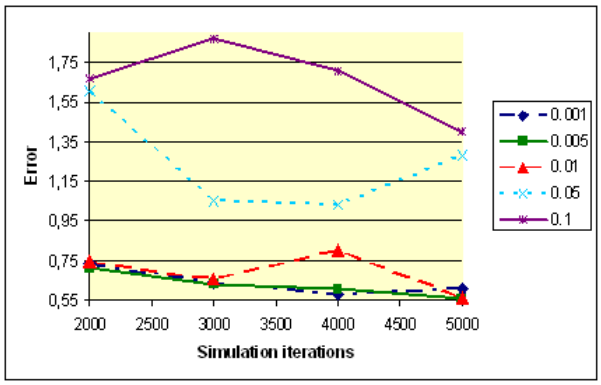



Figure 5: Results for Munin2 with $\delta = 0.1, 0.05, 0.01, 0.005$ and $0.001$.

also arises when using tree pruning as approximation method, since this former method is good to capture uniform regions.

Regarding the high variance associated with the estimations suggested by the graphs in figures 3 and 5, it can be caused by multiple factorisations of the same tree: A tree can be factorised for some variable and afterwards, one of the resulting factors, can be factorised again when deleting another variable. In this case, even if the local error is controlled by $\delta$, there is a global error due to the concatenations of factorisations that is difficult to handle. Perhaps the solution is to forbid that trees that result from a factorisations be factorised again.

## 6 Conclusions

In this paper we have introduced a new version of the dynamic importance sampling algorithm proposed in (Moral and Salmerón, 2005). The novelty consists of using the approximate factorisation of probability trees to control the approximation of the sampling distributions and, in consequence, the final approximations.

The experimental results suggest that the method is valid but perhaps more difficult to control than tree pruning.

Even though the goal of this paper was to analyse the performance of tree factorisation as a stand-alone approximation method, one immediate conclusion that can be drawn from the work presented here is that the joint use of approximate factorisation and tree pruning should be studied. We have carried out some experiment in this direction and the first results are very promising: the variance in the estimations and the computing time seems to be significantly reduced. The main problem of combining both methods is that the difficulty of controlling the final error of the estimation through the parameters of the algorithm increases. We leave for an extended version of this paper a detailed insight on this issue. The experiments we are currently carrying out suggest that the most sensible way of combining both approximation procedures is to use slight tree pruning and also low values for the tolerance of the factorisation. Otherwise, the error introduced by one of the approximation method can affect the other.

It seems difficult to establish a general rule for determining which approximation method is preferable. Depending on the network and the propagation algorithm, the performance of both techniques may vary. For instance, approximate factorisation is appropriate for algorithms that handle factorised potentials, as Lazy propagation or the dynamic importance sampling method used in this paper.

Finally, an efficiency comparison between the resulting algorithm and the dynamic importance sampling algorithm in (Moral and Salmerón, 2005) will be one of the issues that we plan to aim at.

### Acknowledgments

This work has been supported by the Spanish Ministry of Education and Science through project TIN2004-06204-C03-01.

## References

S. Andreassen, F.V. Jensen, S.K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A.R. Sorensen, A. Rosenfalck, and F. Jensen, 1989. *Computer-Aided Electromyography and Expert Systems*, chapter MUNIN - an expert EMG assistant, pages 255–277. Elsevier Science Publishers B.V., Amsterdam.

A. Cano, S. Moral, and A. Salmerón. 2000. Penniless propagation in join trees. *International Journal of Intelligent Systems*, 15:1027–1059.

A. Cano, S. Moral, and A. Salmerón. 2002. Lazy evaluation in Penniless propagation over join trees. *Networks*, 39:175–185.

A. Cano, S. Moral, and A. Salmerón. 2003. Novel strategies to approximate probability trees in Penniless propagation. *International Journal of Intelligent Systems*, 18:193–203.

J. Cheng and M.J. Druzdzel. 2000. AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research*, 13:155–188.

P. Dagum and M. Luby. 1993. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153.

Elvira Consortium. 2002. Elvira: An environment for creating and using probabilistic graphical models. In J.A. Gmez and A. Salmern, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models*, pages 222–230.

K.W. Fertig and N.R. Mann. 1980. An accurate approximation to the sampling distribution of the studentized extreme-valued statistic. *Technometrics*, 22:83–90.

L.D. Hernández, S. Moral, and A. Salmerón. 1998. A Monte Carlo algorithm for probabilistic propagation in belief networks based on importance sampling and stratified simulation techniques. *International Journal of Approximate Reasoning*, 18:53–91.

F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. 1990. Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, 4:269–282.

C.S. Jensen, A. Kong, and U. Kjærulff. 1995. Blocking Gibbs sampling in very large probabilistic expert systems. *International Journal of Human-Computer Studies*, 42:647–666.

U. Kjærulff. 1994. Reduction of computational complexity in Bayesian networks through removal of weak dependencies. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 374–382. Morgan Kaufmann, San Francisco.

I. Martínez, S. Moral, C. Rodríguez, and A. Salmerón. 2002. Factorisation of probability trees and its application to inference in Bayesian networks. In J.A. Gámez and A. Salmerón, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models*, pages 127–134.

I. Martínez, S. Moral, C. Rodríguez, and A. Salmerón. 2005. Approximate factorisation of probability trees. *ECSQARU'05. Lecture Notes in Artificial Intelligence*, 3571:51–62.

S. Moral and A. Salmerón. 2005. Dynamic importance sampling in Bayesian networks based on probability trees. *International Journal of Approximate Reasoning*, 38(3):245–261.

J. Pearl. 1987. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32:247–257.

R.Y. Rubinstein. 1981. *Simulation and the Monte Carlo Method*. Wiley (New York).

A. Salmerón, A. Cano, and S. Moral. 2000. Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413.

R.D. Shachter and M.A. Peot. 1990. Simulation approaches to general probabilistic inference on belief networks. In M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, volume 5, pages 221–231. North Holland (Amsterdam).

N.L. Zhang and D. Poole. 1996. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328.

# Learning Semi-Markovian Causal Models using Experiments

Stijn Meganck[1], Sam Maes[2], Philippe Leray[2] and Bernard Manderick[1]
[1]CoMo
Vrije Universiteit Brussel
Brussels, Belgium
[2]LITIS
INSA Rouen
St. Etienne du Rouvray, France

**Abstract**

Semi-Markovian causal models (SMCMs) are an extension of causal Bayesian networks for modeling problems with latent variables. However, there is a big gap between the SMCMs used in theoretical studies and the models that can be learned from observational data alone. The result of standard algorithms for learning from observations, is a complete partially ancestral graph (CPAG), representing the Markov equivalence class of maximal ancestral graphs (MAGs). In MAGs not all edges can be interpreted as immediate causal relationships. In order to apply state-of-the-art causal inference techniques we need to completely orient the learned CPAG and to transform the result into a SMCM by removing non-causal edges. In this paper we combine recent work on MAG structure learning from observational data with causal learning from experiments in order to achieve that goal. More specifically, we provide a set of rules that indicate which experiments are needed in order to transform a CPAG to a completely oriented SMCM and how the results of these experiments have to be processed. We will propose an alternative representation for SMCMs that can easily be parametrised and where the parameters can be learned with classical methods. Finally, we show how this parametrisation can be used to develop methods to efficiently perform both probabilistic and causal inference.

## 1 Introduction

This paper discusses graphical models that can handle latent variables without explicitly modeling them quantitatively. For such problem domains, several paradigms exist, such as *semi-Markovian causal models* or *maximal ancestral graphs*. Applying these techniques to a problem domain consists of several steps, typically: structure learning from observational and experimental data, parameter learning, probabilistic inference, and, quantitative causal inference.

A problem is that each existing approach only focuses on one or a few of all the steps involved in the process of modeling a problem including latent variables. The goal of this paper is to investigate the integral process from learning from observational and experimental data unto different types of efficient inference.

Semi-Markovian causal models (SMCMs) (Pearl, 2000; Tian and Pearl, 2002) are specifically suited for performing quantitative causal inference in the presence of latent variables. However, at this time no efficient parametrisation of such models is provided and there are no techniques for performing efficient probabilistic inference. Furthermore there are no techniques for learning these models from data issued from observations, experiments or both.

Maximal ancestral graphs (MAGs), developed in (Richardson and Spirtes, 2002) are specifically suited for structure learning from observational data. In MAGs every edge depicts an ancestral relationship. However, the techniques only learn up to Markov equivalence

and provide no clues on which additional experiments to perform in order to obtain the fully oriented causal graph. See (Eberhardt et al., 2005; Meganck et al., 2006) for that type of results on Bayesian networks without latent variables. Furthermore, no parametrisation for discrete variables is provided for MAGs (only one in terms of Gaussian distributions) and no techniques for probabilistic and causal inference have been developed.

We have chosen to use SMCMs in this paper, because they are the only formalism that allows to perform causal inference while taking into account the influence of latent variables. However, we will combine existing techniques to learn MAGs with newly developed methods to provide an integral approach that uses both observational data and experiments in order to learn fully oriented semi-Markovian causal models.

In this paper we also introduce an alternative representation for SMCMs together with a parametrisation for this representation, where the parameters can be learned from data with classical techniques. Finally, we discuss how probabilistic and quantitative causal inference can be performed in these models.

The next section introduces the necessary notations and definitions. It also discusses the semantical and other differences between SMCMs and MAGs. In section 3, we discuss structure learning for SMCMs. Then we introduce a new representation for SMCMs that can easily be parametrised. We also show how both probabilistic and causal inference can be performed with the help of this new representation.

## 2 Notation and Definitions

We start this section by introducing notations and defining concepts necessary in the rest of this paper. We will also clarify the differences and similarities between the semantics of SMCMs and MAGs.

### 2.1 Notation

In this work uppercase letters are used to represent variables or sets of variables, i.e., $V = \{V_1, \ldots, V_n\}$, while corresponding lowercase letters are used to represent their instantiations, i.e., $v_1, v_2$ and $v$ is an instantiation of all $v_i$. $P(V_i)$ is used to denote the probability distribution over all possible values of variable $V_i$, while $P(V_i = v_i)$ is used to denote the probability distribution over the instantiation of variable $V_i$ to value $v_i$. Usually, $P(v_i)$ is used as an abbreviation of $P(V_i = v_i)$.

The operators $Pa(V_i), Anc(V_i), Ne(V_i)$ denote the observable parents, ancestors and neighbors respectively of variable $V_i$ in a graph and $Pa(v_i)$ represents the values of the parents of $V_i$. Likewise, the operator $LPa(V_i)$ represents the latent parents of variable $V_i$. If $V_i \leftrightarrow V_j$ appears in a graph then we say that they are spouses, i.e., $V_i \in Sp(V_j)$ and vice versa.

When two variables $V_i, V_j$ are independent we denote it by $(V_i \perp\!\!\!\perp V_j)$, when they are dependent by $(V_i \not\perp\!\!\!\perp V_j)$.

### 2.2 Semi-Markovian Causal Models

Consider the model in Figure 1(a), it is a problem with observable variables $V_1, \ldots, V_6$ and latent variables $L_1, L_2$ and it is represented by a directed acyclic graph (DAG). As this DAG represents the actual problem, henceforth we will refer to it as the **underlying DAG**.

The central graphical modeling representation that we use are the semi-Markovian causal models (Tian and Pearl, 2002).

**Definition 1.** *A **semi-Markovian causal model** (SMCM) is a representation of a causal Bayesian network (CBN) with observable variables $V = \{V_1, \ldots, V_n\}$ and latent variables $L = \{L_1, \ldots, L_{n'}\}$. Furthermore, every latent variable has no parents (i.e., is a root node) and has exactly two children that are both observed.*

See Figure 1(b) for an example SMCM representing the underlying DAG in (a).

In a SMCM each directed edge represents an immediate autonomous causal relation between the corresponding variables. Our operational definition of causality is as follows: a relation from variable $C$ to variable $E$ is causal in a certain context, when a manipulation in the form of a randomised controlled experiment on vari-

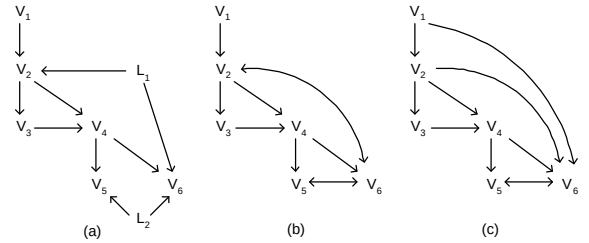



Figure 1: (a) A problem domain represented by a causal DAG model with observable and latent variables. (b) A semi-Markovian causal model representation of (a). (c) A maximal ancestral graph representation of (a).

able $C$, induces a change in the probability distribution of variable $E$, in that specific context (Neapolitan, 2003).

In a SMCM a bi-directed edge between two variables represents a latent variable that is a common cause of these two variables. Although this seems very restrictive, it has been shown that models with arbitrary latent variables can be converted into SMCMs while preserving the same independence relations between the observable variables (Tian and Pearl, 2002).

The semantics of both directed and bi-directed edges imply that SMCMs are not maximal. This means that they do not represent all dependencies between variables with an edge. This is because in a SMCM an edge either represents an immediate causal relation or a latent common cause, and therefore dependencies due to a so called inducing path, will not be represented by an edge.

A node is a collider on a path if both its immediate neighbors on the path point into it.

**Definition 2.** *An **inducing path** is a path in a graph such that each observable non-endpoint node of the path is a collider, and an ancestor of at least one of the endpoints.*

Inducing paths have the property that their endpoints can not be separated by conditioning on any subset of the observable variables. For instance, in Figure 1(a), the path $V_1 \rightarrow V_2 \leftarrow L_1 \rightarrow V_6$ is inducing.

SMCMs are specifically suited for another type of inference, i.e., causal inference.

**Definition 3.** ***Causal inference** is the process of calculating the effect of manipulating some variables $X$ on the probability distribution of some other variables $Y$; this is denoted as $P(Y = y|do(X = x))$.*

An example causal inference query in the SMCM of Figure 1(b) is $P(V_6 = v_6|do(V_2 = v_2))$.

Tian and Pearl have introduced theoretical causal inference algorithms to perform causal inference in SMCMs (Pearl, 2000; Tian and Pearl, 2002). However these algorithms assume that any distribution that can be obtained from the JPD over the observable variables is available. We will show that their algorithm performs efficiently on our parametrisation.

### 2.3 Maximal Ancestral Graphs

Maximal ancestral graphs are another approach to modeling with latent variables (Richardson and Spirtes, 2002). The main research focus in that area lies on learning the structure of these models.

Ancestral graphs (AGs) are complete under marginalisation and conditioning. We will only discuss AGs without conditioning as is commonly done in recent work.

**Definition 4.** *An **ancestral graph** without conditioning is a graph containing directed $\rightarrow$ and bi-directed $\leftrightarrow$ edges, such that there is no bi-directed edge between two variables that are connected by a directed path.*

Pearl's $d$-separation criterion can be extended to be applied to ancestral graphs, and is called $m$-separation in this setting. $M$-separation characterizes the independence relations represented by an ancestral graph.

**Definition 5.** *An ancestral graph is said to be **maximal** if, for every pair of non-adjacent nodes $(V_i, V_j)$ there exists a set $Z$ such that $V_i$ and $V_j$ are independent conditional on $Z$.*

A non-maximal AG can be transformed into a MAG by adding some bi-directed edges (indicating confounding) to the model. See Figure 1(c) for an example MAG representing the same model as the underlying DAG in (a).

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

Bi-directed edges represent a latent common cause between the variables. However, if there is a latent common cause between two variables $A$ and $B$, and there is also a directed path between $A$ and $B$ in the underlying DAG, then in the MAG the ancestral relation takes precedence and a directed edge will be found between the variables. $V_2 \rightarrow V_6$ in Figure 1(c) is an example of such an edge.

Furthermore, as MAGs are maximal, there will also be edges between variables that have no immediate connection in the underlying DAG, but that are connected via an inducing path. The edge $V_1 \rightarrow V_6$ in Figure 1(c) is an example of such an edge.

These semantics of edges make causal inference in MAGs virtually impossible. As stated by the *Manipulation Theorem* (Spirtes et al., 2000), in order to calculate the causal effect of a variable $A$ on another variable $B$, the immediate parents (i.e., the pre-intervention causes) of $A$ have to be removed from the model. However, as opposed to SMCMs, in MAGs an edge does not necessarily represent an immediate causal relationship, but rather an ancestral relationship and hence in general the modeler does not know which are the real immediate causes of a manipulated variable.

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

*Complete partial ancestral graphs* (CPAGs) are defined in (Zhang and Spirtes, 2005) in the following way.

**Definition 6.** *Let $[G]$ be the Markov equivalence class for an arbitrary MAG $G$. The **complete partial ancestral graph** (CPAG) for $[G]$, $P_G$, is a graph with possibly the following edges $\rightarrow, \leftrightarrow, o\!\!-\!\!o, o\!\!\rightarrow$, such that*

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

### 2.4 Assumptions

As is customary in the graphical modeling research area, the SMCMs we take into account in this paper are subject to some simplifying assumptions:

1. *Stability*, i.e., the independencies in the CBN with observed and latent variables that generates the data are structural and not due to several influences exactly canceling each other out (Pearl, 2000).

2. Only a *single immediate connection* per two variables in the underlying DAG. I.e., we do not take into account problems where two variables that are connected by an immediate causal edge are also confounded by a latent variable causing both variables. Constraint based learning techniques such as IC* (Pearl, 2000) and FCI (Spirtes et al., 2000) also do not explicitly recognise multiple edges between variables. However, in (Tian and Pearl, 2002) Tian presents an algorithm for performing causal inference where such relations between variables are taken into account.

3. *No selection bias.* Mimicking recent work, we do not take into account latent variables that are conditioned upon, as can be the consequence of selection effects.

4. *Discrete variables.* All the variables in our models are discrete.

## 3 Structure learning

Just as learning a graphical model in general, learning a SMCM consists of two parts: structure learning and parameter learning. Both can be done using data, expert knowledge and/or experiments. In this section we discuss only structure learning.

### 3.1 Without latent variables

Learning the structure of Bayesian networks without latent variables has been studied by a number of researchers (Pearl, 2000; Spirtes et al., 2000). The results of applying one of those algorithms is a representative of the Markov equivalence class.

In order to perform probabilistic or causal inference, we need a fully oriented structure. For probabilistic inference this can be any representative of the Markov equivalence class, but for causal inference we need the correct causal graph that models the underlying system. In order to achieve this, additional experiments have to be performed.

In previous work (Meganck et al., 2006), we studied learning the completely oriented structure for causal Bayesian networks without latent variables. We proposed a solution to minimise the total cost of the experiments needed by using elements from decision theory. The techniques used there could be extended to the results of this paper.

### 3.2 With latent variables

In order to learn graphical models with latent variables from observational data, the Fast Causal Inference (FCI) algorithm (Spirtes et al., 2000) has been proposed. Recently this result has been extended with the complete tail augmentation rules introduced in (Zhang and Spirtes, 2005). The results of this algorithm is a CPAG, representing the Markov equivalence class of MAGs consistent with the data.

As mentioned above for MAGs, in a CPAG the directed edges have to be interpreted as being ancestral instead of causal. This means that there is a directed edge from $V_i$ to $V_j$ if $V_i$ is an ancestor of $V_j$ in the underlying DAG

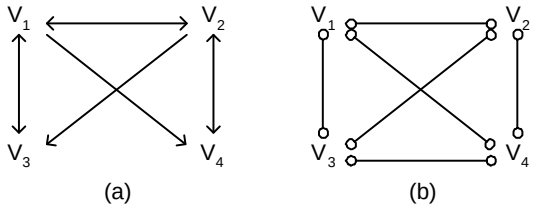



Figure 2: (a) A SMCM. (b) Result of FCI, with an i-false edge $V_3 o\!\!-\!\!o V_4$.

and there is no subset of observable variables $D$ such that $(V_i \perp\!\!\!\perp V_j | D)$. This does not necessarily mean that $V_i$ has an immediate causal influence on $V_j$: it may also be a result of an inducing path between $V_i$ and $V_j$. For instance in Figure 1(c), the link between $V_1$ and $V_6$ is present due to the inducing path $V_1, V_2, L_1, V_6$ shown in Figure 1(a).

Inducing paths may also introduce an edge of type $o\!\!\rightarrow$ or $o\!\!-\!\!o$ between two variables, indicating either a directed or bi-directed edge, although there is no immediate influence in the form of an immediate causal influence or latent common cause between the two variables. An example of such a link is $V_3 o\!\!-\!\!o V_4$ in Figure 2.

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

### 3.3 Transforming the CPAG

Our goal is to transform a given CPAG in order to obtain a SMCM that corresponds to the correspondig DAG. Remember that in general there are three types of edges in a CPAG: $\rightarrow$, $o\!\!\rightarrow$, $o\!\!-\!\!o$, in which $o$ means either a tail mark $-$ or a directed mark $>$. So one of the tasks to

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

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

### 3.3.1 Performing experiments

The experiments discussed here play the role of the manipulations that define a causal relation (cf. Section 2.2). An experiment on a variable $V_i$, i.e., a randomised controlled experiment, removes the influence of other variables in the system on $V_i$. The experiment forces a distribution on $V_i$, and thereby changes the joint distribution of all variables in the system that depend directly or indirectly on $V_i$ but does not change the conditional distribution of other variables given values of $V_i$. After the randomisation, the associations of the remaining variables with $V_i$ provide information about which variables are influenced by $V_i$ (Neapolitan, 2003). When we perform the experiment we cut all influence of other variables on $V_i$. Graphically this corresponds to removing all incoming edges into $V_i$ from the underlying DAG.

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

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

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

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

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

First we have to introduce the following definition:

**Definition 7.** *A **potentially directed path** (p.d. path) in a CPAG is a path made only of edges of types $o\!\!\rightarrow$ and $\rightarrow$, with all arrowheads in the same direction. A p.d. path from $V_i$ to $V_j$ is denoted as $V_i \dashrightarrow V_j$.*

### 3.3.2 Solving $o\!\!\rightarrow$

An overview of the different rules for solving $o\!\!\rightarrow$ is given in Table 1

| | |
|---|---|
| Type 1(a)<br>Given | <br>$Ao\!\!\rightarrow B$<br>$exp(A) \not\rightsquigarrow B$ |
| Action | $A \leftrightarrow B$ |
| Type 1(b)<br>Given | <br>$Ao\!\!\rightarrow B$<br>$exp(A) \rightsquigarrow B$<br>$\nexists$p.d. path $(length \geq 2) A \dashrightarrow B$ |
| Action | $A \rightarrow B$ |
| Type 1(c)<br>Given | <br>$Ao\!\!\rightarrow B$<br>$exp(A) \rightsquigarrow B$<br>$\exists$p.d. path $(length \geq 2) A \dashrightarrow B$ |
| Action | Block all p.d. paths by conditioning on blocking set $D$<br>(a) $exp(A)\|D \rightsquigarrow B$: $A \rightarrow B$<br>(b) $exp(A)\|D \not\rightsquigarrow B$: $A \leftrightarrow B$ |

Table 1: An overview of the different actions needed to complete edges of type $o\!\!\rightarrow$.

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

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

If $exp(V_i) \rightsquigarrow V_j$, then we know for sure that there is an influence of $V_i$ on $V_j$, we now need to discover whether this influence is immediate or via some intermediate variables. Therefore we make a difference whether there is a potentially directed (p.d.) path between $V_i$ and $V_j$ of length $\geq 2$, or not. If no such path exists, then the influence has to be immediate and the edge is found $V_i \rightarrow V_j$ (Type 1(b)).

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

### 3.3.3 Solving *o–o*

An overview of the different rules for solving *o–o* is given in Table 2.

For any edge $V_i o\!\!-\!\!o V_j$, we have no information at all, so we might need to perform experiments on both variables.

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

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

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

| | |
|---|---|
| Type 2(a)<br>Given | <br>$Ao\!\!-\!\!oB$<br>$exp(A) \not\rightsquigarrow B$ |
| Action | $A \leftarrow\!\!oB$(⇒Type 1) |
| Type 2(b)<br>Given | <br>$Ao\!\!-\!\!oB$<br>$exp(A) \rightsquigarrow B$<br>∄p.d. path $(length \geq 2) A \dashrightarrow B$ |
| Action | $A \rightarrow B$ |
| Type 2(c)<br>Given | <br>$Ao\!\!-\!\!oB$<br>$exp(A) \rightsquigarrow B$<br>∃p.d. path $(length \geq 2) A \dashrightarrow B$ |
| Action | Block all p.d. paths by conditioning on blocking set $D$<br>(a) $exp(A)\|D \rightsquigarrow B$: $A \rightarrow B$<br>(b) $exp(A)\|D \not\rightsquigarrow B$: $A \leftarrow\!\!oB$<br>(⇒Type 1) |

Table 2: An overview of the different actions needed to complete edges of type *o–o*.

### 3.3.4 Removing inducing path edges

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

For instance, in Figure 1(a) the path $V_1, V_2, L_1, V_6$ is an inducing path, which causes the FCI algorithm to find an i-false edge between $V_1$ and $V_6$, see Figure 1(c). Another example is given in Figure 2 where the SMCM is given in (a) and the result of FCI in (b). The

edge between $V_3$ and $V_4$ in (b) is a consequence of the inducing path via the observable variables $V_3, V_1, V_2, V_4$.

In order to be able to apply a causal inference algorithm we need to remove all i-false edges from the learned structure. We need to identify the substructures that can indicate this type of edges. This is easily done by looking at any two variables that are connected by an immediate connection, and when this edge is removed, they have at least one inducing path between them. To check whether the immediate connection needs to be present we have to block all inducing paths by performing one or more experiments on an inducing path blocking set (i-blocking set) $D^{ip}$ and block all other paths by conditioning on a blocking set $D$. If $V_i$ and $V_j$ are dependent, i.e., $V_i \not\perp\!\!\!\perp V_j$ under these circumstances the edge is correct and otherwise it can be removed.

In the example of Figure 1(c), we can block the inducing path by performing an experiment on $V_2$, and hence can check that $V_1$ and $V_6$ do not covary with each other in these circumstances, so the edge can be removed.

In Table 3 an overview of the actions to resolve i-false edges is given.

| | |
|---|---|
| Given | A pair of connected variables $V_i, V_j$<br>A set of inducing paths $V_i, \ldots, V_j$ |
| Action | Block all inducing paths $V_i, \ldots, V_j$ by conditioning on i-blocking set $D^{ip}$.<br>Block all other paths between $V_i$ and $V_j$ by conditioning on blocking set $D$.<br>When performing all $exp(D^{ip})\|D$:<br>if $V_i \not\perp\!\!\!\perp V_j$: confounding is real<br>else remove edge between $V_i$, $V_j$ |

Table 3: Removing inducing path edges.

### 3.4 Example

We will demonstrate a number of steps to discover the completely oriented SMCM (Figure 1(b)) based on the result of the FCI algorithm applied on observational data generated from the underlying DAG in Figure 1(a). The result

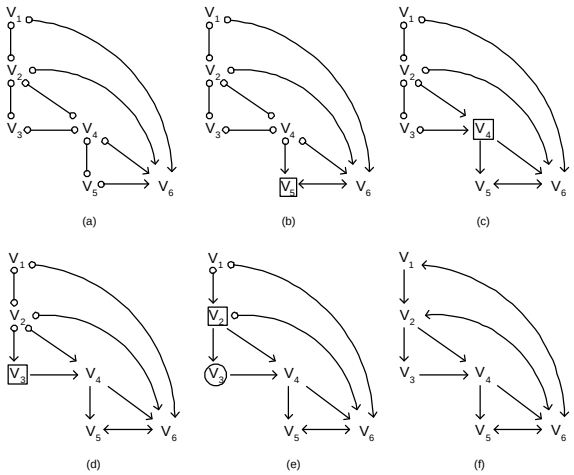



Figure 3: (a) The result of FCI on data of the underlying DAG of Figure 1(a). (b) Result of an experiment on $V_5$. (c) After experiment on $V_4$. (d) After experiment on $V_3$. (e) After experiment on $V_2$ while conditioning on $V_3$. (f) After resolving all problems of Type 1 and 2.

of the FCI algorithm can be seen in Figure 3(a). We will first resolve problems of Type 1 and 2, and then remove i-false edges. The result of each step is indicated in Figure 3.

- $exp(V_5)$
  - $V_5 o\!\!-\!\!o V_4$:
    $exp(V_5) \not\leftrightarrow V_4 \Rightarrow V_4 o\!\!\rightarrow V_5$ (Type 2(a))
  - $V_5 o\!\!\rightarrow V_6$:
    $exp(V_5) \not\leftrightarrow V_6 \Rightarrow V_5 \leftrightarrow V_6$ (Type 1(a))
- $exp(V_4)$
  - $V_4 o\!\!-\!\!o V_2$:
    $exp(V_4) \not\leftrightarrow V_2 \Rightarrow V_2 o\!\!\rightarrow V_4$ (Type 2(a))
  - $V_4 o\!\!-\!\!o V_3$:
    $exp(V_4) \not\leftrightarrow V_3 \Rightarrow V_3 o\!\!\rightarrow V_4$ (Type 2(a))
  - $V_4 o\!\!\rightarrow V_5$:
    $exp(V_4) \rightsquigarrow V_5 \Rightarrow V_4 \rightarrow V_5$ (Type 1(b))
  - $V_4 o\!\!\rightarrow V_6$:
    $exp(V_4) \rightsquigarrow V_6 \Rightarrow V_4 \rightarrow V_6$ (Type 1(b))
- $exp(V_3)$
  - $V_3 o\!\!-\!\!o V_2$:
    $exp(V_3) \not\leftrightarrow V_2 \Rightarrow V_2 o\!\!\rightarrow V_3$ (Type 2(a))
  - $V_3 o\!\!\rightarrow V_4$:
    $exp(V_3) \rightsquigarrow V_4 \Rightarrow V_3 \rightarrow V_4$ (Type 1(b))

- $exp(V_2)$ and $exp(V_2)|V_3$, because two p.d. paths between $V_2$ and $V_4$
  - $V_2 o\!\!-\!\!o V_1$: $exp(V_2) \not\rightsquigarrow V_1 \Rightarrow V_1 o\!\!\rightarrow V_2$ (Type 2(a))
  - $V_2 o\!\!\rightarrow V_3$: $exp(V_2) \rightsquigarrow V_3 \Rightarrow V_2 \rightarrow V_3$ (Type 1(b))
  - $V_2 o\!\!\rightarrow V_4$: $exp(V_2)|V_3 \rightsquigarrow V_4 \Rightarrow V_2 \rightarrow V_4$ (Type 1(c))

After resolving all problems of Type 1 and 2 we end up with the SMCM structure shown in Figure 3(f). This representation is no longer consistent with the MAG representation since there are bi-directed edges between two variables on a directed path, i.e., $V_2, V_6$. There is a potentially i-false edge $V_1 \leftrightarrow V_6$ in the structure with inducing path $V_1, V_2, V_6$, so we need to perform an experiment on $V_2$, blocking all other paths between $V_1$ and $V_6$ (this is also done by $exp(V_2)$ in this case). Given that the original structure is as in Figure 1(a), performing $exp(V_2)$ shows that $V_1$ and $V_6$ are independent, i.e., $exp(V_2) : (V_1 \perp\!\!\!\perp V_6)$. Thus the bi-directed edge between $V_1$ and $V_6$ is removed, giving us the SMCM of Figure 1(b).

## 4 Parametrisation of SMCMs

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

We start this section by discussing the factorisation for SMCMs introduced in (Tian and Pearl, 2002). From that result we derive an additional representation for SMCMs and a parametrisation that facilitates probabilistic and causal inference. We will also discuss how these parameters can be learned from data.

### 4.1 Factorising with Latent Variables

Consider an underlying DAG with observable variables $V = \{V_1, \ldots, V_n\}$ and latent variables $L = \{L_1, \ldots, L_{n'}\}$. Then the joint probability distribution can be written as the following mixture of products:

$$P(v) = \sum_{\{l_k | L_k \in L\}} \prod_{V_i \in V} P(v_i | Pa(v_i), LPa(v_i)) \prod_{L_j \in L} P(l_j). \quad (1)$$

Taking into account that in a SMCM the latent variables are implicitly represented by bi-directed edges, we introduce the following definition.

**Definition 8.** *In a SMCM, the set of observable variables can be partitioned by assigning two variables to the same group iff they are connected by a bi-directed path. We call such a group a* ***c-component*** *(from "confounded component") (Tian and Pearl, 2002).*

For instance, in Figure 1(b) variables $V_2, V_5, V_6$ belong to the same c-component. Then it can be readily seen that c-components and their associated latent variables form respective partitions of the observable and latent variables. Let $Q[S_i]$ denote the contribution of a c-component with observable variables $S_i \subset V$ to the mixture of products in Equation 1. Then we can rewrite the JPD as follows: $P(v) = \prod_{i \in \{1,\ldots,k\}} Q[S_i]$.

Finally, (Tian and Pearl, 2002) proved that each $Q[S]$ could be calculated as follows. Let $V_{o_1} < \ldots < V_{o_n}$ be a topological order over $V$, and let $V^{(i)} = V_{o_1} < \ldots < V_{o_i}$, $i = 1, \ldots, n$, and $V^{(0)} = \emptyset$.

$$Q[S] = \prod_{V_i \in S} P(v_i | (T_i \cup Pa(T_i)) \backslash \{V_i\}) \quad (2)$$

where $T_i$ is the c-component of the SMCM $G$ reduced to variables $V^{(i)}$, that contains $V_i$. The SMCM $G$ reduced to a set of variables $V' \subset V$ is the graph obtained by removing from the graph all variables $V \backslash V'$ and the edges that are connected to them.

In the rest of this section we will develop a method for deriving a DAG from a SMCM.

We will show that the classical factorisation $\prod P(v_i|Pa(v_i))$ associated with this DAG, is the same as the one associated with the SMCM, as above.

### 4.2 Parametrised representation

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

#### 4.2.1 PR-representation

Consider $V_{o_1} < \ldots < V_{o_n}$ to be a topological order $O$ over the observable variables $V$, only considering the directed edges of the SMCM, and let $V^{(i)} = V_{o_1} < \ldots < V_{o_i}$, $i = 1, \ldots, n$, and $V^{(0)} = \emptyset$. Then Table 4 shows how the parametrised (PR-) representation can be obtained from the original SMCM structure.

| Given a SMCM $G$ and a topological ordering $O$, the PR-representation has these properties: |
|---|
| 1. The nodes are $V$, i.e., the observable variables of the SMCM. |
| 2. The directed edges are the same as in the SMCM. |
| 3. The confounded edges are replaced by a number of directed edges as follows: Add an edge from node $V_i$ to node $V_j$ iff: a) $V_i \in (T_j \cup Pa(T_j))$, where $T_j$ is the c-component of $G$ reduced to variables $V^{(j)}$ that contains $V_j$, b) and there was not already an edge between nodes $V_i$ and $V_j$. |

Table 4: Obtaining the PR-representation.

This way each variable becomes a child of the variables it would condition on in the calculation of the contribution of its c-component as in Equation (2).

Figure 4(a) shows the PR-representation of the SMCM in Figure 1(a). The topological order that was used here is $V_1 < V_2 < V_3 < V_4 < V_5 < V_6$ and the directed edges that have been added are $V_1 \rightarrow V_5$, $V_2 \rightarrow V_5$, $V_1 \rightarrow V_6$, $V_2 \rightarrow V_6$, and, $V_5 \rightarrow V_6$.

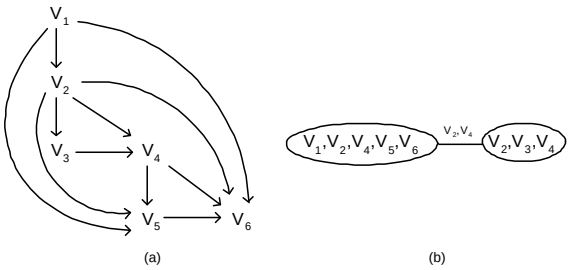



Figure 4: (a) The PR-representation applied to the SMCM of Figure 1(b). (b) Junction tree representation of the DAG in (a).

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

**Theorem 9.** *The PR-representation PR derived from a SMCM S is an* ***I-map*** *of that SMCM.*

*Proof.* Consider two variables $A$ and $B$ in $PR$ that are not connected by an edge. This means that they are not independent, since a necessary condition for two variables to be conditionally independent in a stable DAG is that they are not connected by an edge. $PR$ is stable as we consider only stable problems (Assumption 1 in Section 2.4).

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

#### 4.2.2 Parametrisation

For this DAG we can use the same parametrisation as for classical BNs, i.e., learning $P(v_i|Pa(v_i))$ for each variable, where $Pa(v_i)$ denotes the parents in the new DAG. In this way the JPD over the observable variables factorises as in a classical BN, i.e., $P(v) = \prod P(v_i|Pa(v_i))$. This follows immediately from the definition of a $c$-component and from Equation (2).

#### 4.2.3 Optimising the Parametrisation

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

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

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

#### 4.2.4 Learning parameters

As the PR-representation of SMCMs is a DAG as in the classical Bayesian network formalism, the parameters that have to be learned are $P(v_i|Pa(v_i))$. Therefore, techniques such as ML and MAP estimation (Heckerman, 1995) can be applied to perform this task.

### 4.3 Probabilistic inference

One of the most famous existing probabilistic inference algorithm for models without latent variables is the *junction tree* algorithm (JT) (Lauritzen and Spiegelhalter, 1988).

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

We have solved this problem by first transforming the SMCM into its PR-representation, which allows us to apply the junction tree inference algorithm. This is a consequence of the fact that, as previously mentioned, the PR-representation is an $I$-map over the observable variables. And as the JT algorithm is based only on independencies in the DAG, applying it to an $I$-map of the problem gives correct results. See Figure 4(b) for the junction tree obtained from the parametrised representation in Figure 4(a).

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

### 4.4 Causal inference

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

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

By definition in the PR-representation, the parents of each variable are exactly those variables that have to be conditioned on in order to obtain the factor of that variable in the calculation of the $c$-component, see Table 4 and (Tian and Pearl, 2002). Thus, the PR-representation provides all the necessary quantitative information, while the original structure of the SMCM provides the necessary structural information, for the algorithm by Tian and Pearl to be applied.

## 5 Conclusions and Perspectives

In this paper we have proposed a number of solutions to problems that arise when using SMCMs in practice.

More precisely, we showed that there is a big gap between the models that can be learned from data alone and the models that are used in theory. We showed that it is important to retrieve the fully oriented structure of a SMCM,

and discussed how to obtain this from a given CPAG by performing experiments.

For future work we would like to relax the assumptions made in this paper. First of all we want to study the implications of allowing two types of edges between two variables, i.e., confounding as well as a immediate causal relationship. Another direction for possible future work would be to study the effect of allowing multiple joint experiments in other cases than when removing inducing path edges.

Furthermore, we believe that applying the orientation and tail augmentation rules of (Zhang and Spirtes, 2005) after each experiment, might help to reduce the number of experiments needed to fully orient the structure. In this way we could extend to SMCMs our previous results (Meganck et al., 2006) on minimising the total number of experiments in causal models without latent variables. This would allow to compare empirical results with the theoretical bounds developed in (Eberhardt et al., 2005).

Up until now SMCMs have not been parametrised in another way than by the entire joint probability distribution. Another contribution of our paper is that we showed that using an alternative representation, we can parametrise SMCMs in order to perform probabilistic as well as causal inference. Furthermore this new representation allows to learn the parameters using classical methods.

We have informally pointed out that the choice of a topological order when creating the PR-representation, influences its size and thus its efficiency. We would like to investigate this property in a more formal manner. Finally, we have started implementing the techniques introduced in this paper into the structure learning package (SLP)[1] of the Bayesian networks toolbox (BNT)[2] for MATLAB.

## Acknowledgments

This work was partially funded by an IWT-scholarship and by the IST Programme of the European Community, under the PASCAL network of Excellence, IST-2002-506778.

## References

F. Eberhardt, C. Glymour, and R. Scheines. 2005. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among $n$ variables. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 178–183. Edinburgh.

D. Heckerman. 1995. A tutorial on learning with Bayesian networks. Technical report, Microsoft Research.

S. L. Lauritzen and D. J. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, series B*, 50:157–244.

S. Meganck, P. Leray, and B. Manderick. 2006. Learning causal Bayesian networks from observations and experiments: A decision theoretic approach. In *Modeling Decisions in Artificial Intelligence, LNAI 3885*, pages 58 – 69.

R.E. Neapolitan. 2003. *Learning Bayesian Networks*. Prentice Hall.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

J. Pearl. 2000. *Causality: Models, Reasoning and Inference*. MIT Press.

T. Richardson and P. Spirtes. 2002. Ancestral graph Markov models. Technical Report 375, Dept. of Statistics, University of Washington.

T. Richardson and P. Spirtes, 2003. *Causal inference via ancestral graph models*, chapter 3. Oxford Statistical Science Series: Highly Structured Stochastic Systems. Oxford University Press.

P. Spirtes, C. Glymour, and R. Scheines. 2000. *Causation, Prediction and Search*. MIT Press.

J. Tian and J. Pearl. 2002. On the identification of causal effects. Technical Report (R-290-L), UCLA C.S. Lab.

J. Zhang and P. Spirtes. 2005. A characterization of Markov equivalence classes for ancestral graphical models. Technical Report 168, Dept. of Philosophy, Carnegie-Mellon University.

[1] http://banquiseasi.insa-rouen.fr/projects/bnt-slp/
[2] http://bnt.sourceforge.net/

# Geometry of Rank Tests

Jason Morton, Lior Pachter, Anne Shiu, Bernd Sturmfels, and Oliver Wienand
Department of Mathematics, UC Berkeley

**Abstract**

We study partitions of the symmetric group which have desirable geometric properties. The statistical tests defined by such partitions involve counting all permutations in the equivalence classes. These permutations are the linear extensions of partially ordered sets specified by the data. Our methods refine rank tests of non-parametric statistics, such as the sign test and the runs test, and are useful for the exploratory analysis of ordinal data. *Convex rank tests* correspond to probabilistic conditional independence structures known as semi-graphoids. *Submodular rank tests* are classified by the faces of the cone of submodular functions, or by Minkowski summands of the permutohedron. We enumerate all small instances of such rank tests. *Graphical tests* correspond to both graphical models and to graph associahedra, and they have excellent statistical and algorithmic properties.

## 1 Introduction

The non-parametric approach to statistics was introduced by (Pitman, 1937). The emergence of microarray data in molecular biology has led to a number of new tests for identifying significant patterns in gene expression time series; see e.g. (Willbrand, 2005). This application motivated us to develop a mathematical theory of rank tests. We propose that a *rank test* is a partition of $S_n$ induced by a map $\tau : S_n \to T$ from the symmetric group of all permutations of $[n] = \{1, \ldots, n\}$ onto a set $T$ of statistics. The statistic $\tau(\pi)$ is the *signature* of the permutation $\pi \in S_n$. Each rank test defines a partition of $S_n$ into classes, where $\pi$ and $\pi'$ are in the same class if and only if $\tau(\pi) = \tau(\pi')$. We identify $T = \text{image}(\tau)$ with the set of all classes in this partition of $S_n$. Assuming the uniform distribution on $S_n$, the probability of seeing a particular signature $t \in T$ is $1/n!$ times $|\tau^{-1}(t)|$. The computation of a $p$-value for a given permutation $\pi \in S_n$ typically amounts to summing

$$\Pr(\pi') \quad = \quad \frac{1}{n!} \cdot |\, \tau^{-1}\big(\tau(\pi')\big)\,| \qquad (1)$$

over all permutations $\pi'$ with $\Pr(\pi') < \Pr(\pi)$. In Section 2 we explain how existing rank tests can be understood from our point of view.

In Section 3 we describe the class of *convex rank tests* which captures properties of tests used in practice. We work in the language of algebraic combinatorics (Stanley, 1997). Convex rank tests are in bijection with polyhedral fans that coarsen the hyperplane arrangement of $S_n$, and with conditional independence structures known as semi-graphoids (Studený, 2005).

Section 4 is devoted to convex rank tests that are induced by submodular functions. These *submodular rank tests* are in bijection with Minkowski summands of the $(n-1)$-dimensional permutohedron and with structural imset models. Furthermore, these tests are at a suitable level of generality for the biological applications that motivated us. We make the connections to polytopes and independence models concrete by classifying all convex rank tests for $n \leq 5$.

In Section 5 we discuss the class of *graphical tests*. In mathematics, these correspond to graph associahedra, and in statistics to graphical models. The equivalence of these two structures is shown in Theorem 18. The implementation of convex rank tests requires the efficient enumeration of linear extensions of partially ordered sets (posets). A key ingredient is a highly optimized method for computing distributive lattices. Our software is discussed in Section 6.

## 2 Rank tests and posets

A permutation $\pi$ in $S_n$ is a total order on $[n] = \{1, \ldots, n\}$. This means that $\pi$ is a set of $\binom{n}{2}$ ordered pairs of elements in $[n]$. If $\pi$ and $\pi'$ are permutations then $\pi \cap \pi'$ is a partial order.

In the applications we have in mind, the data are vectors $u \in \mathbb{R}^n$ with distinct coordinates. The permutation associated with $u$ is the total order $\pi = \{ (i,j) \in [n] \times [n] \,:\, u_i < u_j \}$. We shall employ two other ways of writing this permutation. The first is the *rank vector* $\rho = (\rho_1, \ldots, \rho_n)$, whose defining properties are $\{\rho_1, \ldots, \rho_n\} = [n]$ and $\rho_i < \rho_j$ if and only if $u_i < u_j$. That is, the coordinate of the rank vector with value $i$ is at the same position as the $i$th smallest coordinate of $u$. The second is the *descent vector* $\delta = (\delta_1, \ldots, \delta_n)$, defined by $u_{\delta_i} > u_{\delta_{i+1}}$. The $i$th coordinate of the descent vector is the position of the $i$th largest value of $u$. For example, if $u = (11, 7, 13)$ then its permutation is represented by $\pi = \{(2,1),(1,3),(2,3)\}$, by $\rho = (2,1,3)$, or by $\delta = (3,1,2)$.

A permutation $\pi$ is a *linear extension* of a partial order $P$ on $[n]$ if $P \subseteq \pi$. We write $\mathcal{L}(P) \subseteq S_n$ for the set of linear extensions of $P$. A partition $\tau$ of the symmetric group $S_n$ is a *pre-convex rank test* if the following axiom holds:

$$(PC) \qquad \begin{array}{c} \text{If } \tau(\pi) = \tau(\pi') \text{ and } \pi'' \in \mathcal{L}(\pi \cap \pi') \\ \text{then } \tau(\pi) = \tau(\pi') = \tau(\pi''). \end{array}$$

Note that $\pi'' \in \mathcal{L}(\pi \cap \pi')$ means $\pi \cap \pi' \subseteq \pi''$. For $n = 3$ the number of all rank tests is the Bell number $B_6 = 203$. Of these 203 set partitions of $S_3$, only 40 satisfy the axiom (PC).

Each class $C$ of a pre-convex rank test $\tau$ corresponds to a poset $P$ on $[n]$; namely, $P$ is the intersection of all total orders in that class: $P = \bigcap_{\pi \in C} \pi$. The axiom (PC) ensures that $C$ coincides with the set $\mathcal{L}(P)$ of all linear extensions of $P$. The inclusion $C \subseteq \mathcal{L}(P)$ is clear. For the reverse inclusion, note that from any permutation $\pi$ in $\mathcal{L}(P)$, we can obtain any other $\pi'$ in $\mathcal{L}(P)$ by a sequence of reversals $(a,b) \mapsto (b,a)$, where each intermediate $\hat{\pi}$ is also in $\mathcal{L}(P)$. Assume $\pi_0 \in \mathcal{L}(P)$ and $\pi_1 \in C$ differ by one reversal $(a,b) \in \pi_0$, $(b,a) \in \pi_1$. Then $(b,a) \notin P$, so there is some $\pi_2 \in C$ such that $(a,b) \in \pi_2$; thus, $\pi_0 \in \mathcal{L}(\pi_1 \cap \pi_2)$ by (PC). This shows $\pi_0 \in C$.

A pre-convex rank test is thus an unordered collection of posets $P_1, , \ldots, P_k$ on $[n]$ that satisfies the property that $S_n$ is the disjoint union of the subsets $\mathcal{L}(P_1), \ldots, \mathcal{L}(P_k)$. The posets $P_i$ that represent the classes in a pre-convex rank test capture the shapes of data vectors.

**Example 1** (The sign test for paired data)**.** The *sign test* is performed on data that are paired as two vectors $u = (u_1, u_2, \ldots, u_m)$ and $v = (v_1, v_2, \ldots, v_m)$. The null hypothesis is that the median of the differences $u_i - v_i$ is 0. The test statistic is the number of differences that are positive. This test is a rank test, because $u$ and $v$ can be transformed into the overall ranks of the $n = 2m$ values, and the rank vector entries can then be compared. This test coarsens the convex rank test which is the MSS of Section 4 with $\mathcal{K} = \{\{1, m+1\}, \{2, m+2\}, \ldots\}$.

**Example 2** (Runs tests)**.** A *runs test* can be used when there is a natural ordering on the data points, such as in a time series. The data are transformed into a sequence of 'pluses' and 'minuses,' and the null hypothesis is that the number of observed runs is no more than that expected by chance. A runs test is a coarsening of the convex rank test $\tau$ described in (Willbrand, 2005, Section 6.1.1) and in Example 4.

These two examples suggest that many tests from classical statistics have a natural refinement by a pre-convex rank test. The term "pre-convex" refers to the following interpretation of the axiom (PC). Consider any two vectors $u$ and $u'$ in $\mathbb{R}^n$, and a convex combination $u'' = \lambda u + (1-\lambda)u'$, with $0 < \lambda < 1$. If $\pi, \pi', \pi''$ are the permutations of $u, u', u''$ then $\pi'' \in \mathcal{L}(\pi \cap \pi')$. Thus the regions in $\mathbb{R}^n$ specified by a pre-convex rank test are convex cones.

## 3 Convex rank tests

A *fan* in $\mathbb{R}^n$ is a finite collection $\mathcal{F}$ of polyhedral cones which satisfies the following properties: (i) if $C \in \mathcal{F}$ and $C'$ is a face of $C$, then $C' \in \mathcal{F}$, (ii) If $C, C' \in \mathcal{F}$, then $C \cap C'$ is a face of $C$. Two vectors $u$ and $v$ in $\mathbb{R}^n$ are *permutation equivalent* when $u_i < u_j$ if and only if

$v_i < v_j$, and $u_i = u_j$ if and only if $v_i = v_j$ for all $i, j \in [n]$. The permutation equivalence classes (of which there are 13 for $n = 3$) induce a fan which we call the $S_n$-*fan*. The maximal cones in the $S_n$-fan, which are the closures of the permutation equivalence classes corresponding to total orders, are indexed by permutations $\delta$ in $S_n$. A *coarsening* of the $S_n$-fan is a fan $\mathcal{F}$ such that every permutation equivalence class of $\mathbb{R}^n$ is fully contained in a cone $C$ of $\mathcal{F}$; $\mathcal{F}$ defines a partition of $S_n$ because each maximal cone of the $S_n$-fan is contained in some cone $C \in \mathcal{F}$. We define a *convex rank test* to be a partition of $S_n$ defined by a coarsening of the $S_n$-fan. We identify the fan with that test.

Two maximal cones of the $S_n$-fan share a *wall* if there exists an index $k$ such that $\delta_k = \delta'_{k+1}$, $\delta_{k+1} = \delta'_k$ and $\delta_i = \delta'_i$ for $i \neq k, k+1$. That is, the corresponding permutations $\delta$ and $\delta'$ differ by an adjacent transposition. To such an unordered pair $\{\delta, \delta'\}$, we associate the following conditional independence (CI) statement:

$$\delta_k \perp\!\!\!\perp \delta_{k+1} \,|\, \{\delta_1, \ldots, \delta_{k-1}\}. \tag{2}$$

This formula defines a map from the set of walls of the $S_n$-fan onto the set of all CI statements

$$\mathcal{T}_n \;=\; \big\{\, i \perp\!\!\!\perp j \,|\, K \;:\; K \subseteq [n]\backslash\{i,j\}\big\}.$$

The map from walls to CI statements is not injective; there are $(n-k-1)!(k-1)!$ walls which are labelled by the statement (2).

Any convex rank test $\mathcal{F}$ is characterized by the collection of walls $\{\delta, \delta'\}$ that are removed when passing from the $S_n$-fan to $\mathcal{F}$. So, from (2), any convex rank test $\mathcal{F}$ maps to a set $\mathcal{M}_\mathcal{F}$ of CI statements corresponding to missing walls. Recall from (Matúš, 2004) and (Studený, 2005) that a subset $\mathcal{M}$ of $\mathcal{T}_n$ is a *semi-graphoid* if the following axiom holds:

$$\begin{aligned} &i \perp\!\!\!\perp j \,|\, K \cup \ell \in \mathcal{M} \text{ and } i \perp\!\!\!\perp \ell \,|\, K \in \mathcal{M} \\ &\text{implies } i \perp\!\!\!\perp j \,|\, K \in \mathcal{M} \text{ and } i \perp\!\!\!\perp \ell \,|\, K \cup j \in \mathcal{M}. \end{aligned}$$

**Theorem 3.** *The map $\mathcal{F} \mapsto \mathcal{M}_\mathcal{F}$ is a bijection between convex rank tests and semi-graphoids.*

**Example 4** (Up-down analysis for $n = 3$)**.** The test in (Willbrand, 2005) is a convex rank test

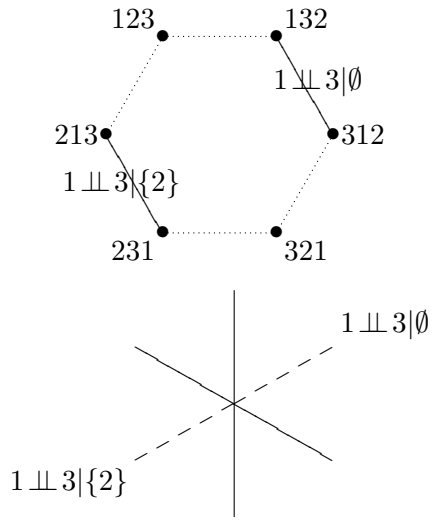



Figure 1: The permutohedron $\mathbf{P}_3$ and the $S_3$-fan projected to the plane. Each permutation is represented by its descent vector $\delta = \delta_1\delta_2\delta_3$.

and is visualized in Figure 1. Permutations are in the same class if they are connected by a solid edge; there are four classes. In the $S_3$-fan, the two missing walls are labeled by conditional independence statements as defined in (2).

**Example 5** (Up-down analysis for $n = 4$)**.** The test $\mathcal{F}$ in (Willbrand, 2005) is shown in Figure 2. The double edges correspond to the 12 CI statements in $\mathcal{M}_\mathcal{F}$. There are 8 classes; e.g., the class $\{3412, 3142, 1342, 1324, 3124\}$ consists of the 5 permutations with up-down pattern $(-,+,-)$.

Our proof of Theorem 3 rests on translating the semi-graphoid axiom for a set of CI statements into geometric statements about the corresponding set of edges of the permutohedron.

The $S_n$-fan is the normal fan (Ziegler, 1995) of the *permutohedron* $\mathbf{P}_n$, which is the convex hull of the vectors $(\rho_1, \ldots, \rho_n) \in \mathbb{R}^n$, where $\rho$ runs over all rank vectors of permutations in $S_n$. The edges of $\mathbf{P}_n$ correspond to walls and are thus labeled with CI statements. A collection of parallel edges of $\mathbf{P}_n$ perpendicular to a hyperplane $x_i = x_j$ corresponds to the set of CI statements $i \perp\!\!\!\perp j|K$, where $K$ ranges over all subsets of $[n]\backslash\{i,j\}$. The two-dimensional faces of $\mathbf{P}_n$ are squares and regular hexagons, and two edges of $\mathbf{P}_n$ have the same label in $\mathcal{T}_n$

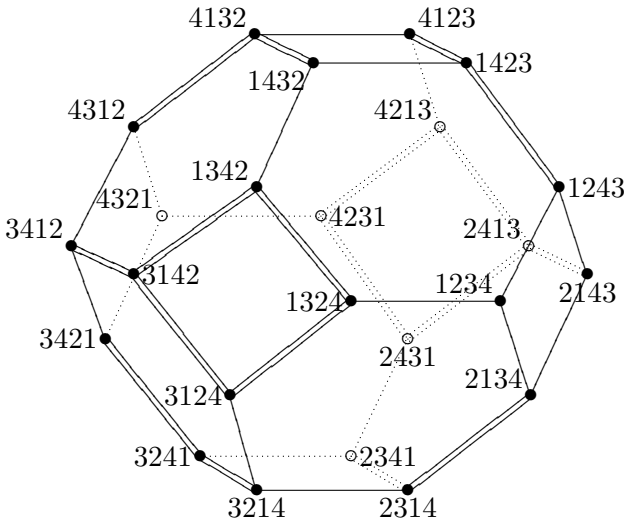



Figure 2: The permutohedron $\mathbf{P}_4$ with vertices marked by descent vectors $\delta$. The test "up-down analysis" is indicated by the double edges.

if, but not only if, they are opposite edges of a square. A semi-graphoid $\mathcal{M}$ can be identified with the set $\mathbf{M}$ of edges with labels from $\mathcal{M}$. The semi-graphoid axiom translates into a geometric condition on the hexagonal faces of $\mathbf{P}_n$.

**Observation 6.** *A set* $\mathbf{M}$ *of edges of the permutohedron* $\mathbf{P}_n$ *is a semi-graphoid if and only if* $\mathbf{M}$ *satisfies the following two axioms:*
**Square axiom:** *Whenever an edge of a square is in* $\mathbf{M}$*, then the opposite edge is also in* $\mathbf{M}$*.*
**Hexagon axiom:** *Whenever two adjacent edges of a hexagon are in* $\mathbf{M}$*, then the two opposite edges of that hexagon are also in* $\mathbf{M}$*.*

Let $\mathbf{M}$ be the subgraph of the edge graph of $\mathbf{P}_n$ defined by the statements in $\mathcal{M}$. Then the classes of the rank test defined by $\mathcal{M}$ are given by the permutations in the path-connected components of $\mathbf{M}$. We regard a path from $\delta$ to $\delta'$ on $\mathbf{P}_n$ as a word $\sigma^{(1)} \cdots \sigma^{(l)}$ in the free associative algebra $\mathcal{A}$ generated by the adjacent transpositions of $[n]$. For example, the word $\sigma_{23} := (23)$ gives the path from $\delta$ to $\delta' = \sigma_{23}\delta = \delta_1\delta_3\delta_2\delta_4 \ldots \delta_n$. The following relations in $\mathcal{A}$ define a presentation of the group algebra of $S_n$:

$(BS)$ $\sigma_{i,i+1}\sigma_{i+k+1,i+k+2} - \sigma_{i+k+1,i+k+2}\sigma_{i,i+1}$,
$(BH)$ $\sigma_{i,i+1}\sigma_{i+1,i+2}\sigma_{i,i+1} - \sigma_{i+1,i+2}\sigma_{i,i+1}\sigma_{i+1,i+2}$,
$(BN)$ $\sigma_{i,i+1}^2 - 1$,

where suitable $i$ and $k$ vary over $[n]$. The first two are the *braid relations*, and the last represents the idempotency of each transposition.

Now, we regard these relations as properties of a set of edges of $\mathbf{P}_n$, by identifying a word and a permutation $\delta$ with the set of edges that comprise the corresponding path in $\mathbf{P}_n$. For example, a set satisfying (BS) is one such that, starting from any $\delta$, the edges of the path $\sigma_{i,i+1}\sigma_{i+k+1,i+k+2}$ are in the set if and only if the edges of the path $\sigma_{i+k+1,i+k+2}\sigma_{i,i+1}$ are in the set. Note then, that (BS) is the square axiom, and (BH) is a weakening of the hexagon axiom of semi-graphoids. That is, implications in either direction hold in a semi-graphoid. However, (BN) holds only directionally in a semi-graphoid: if an edge lies in the semi-graphoid, then its two vertices are in the same class; but the empty path at some vertex $\delta$ certainly does not imply the presence of all incident edges in the semi-graphoid. Thus, for a semi-graphoid, we have (BS) and (BH), but must replace (BN) with the directional version

$$(BN') \qquad \sigma_{i,i+1}^2 \to 1.$$

Consider a path $p$ from $\delta$ to $\delta'$ in a semi-graphoid. A result of (Tits, 1968) gives the following lemma; see also (Brown, 1989, p. 49-51).

**Lemma 7.** *If* $\mathcal{M}$ *is a semi-graphoid, then if* $\delta$ *and* $\delta'$ *lie in the same class of* $\mathcal{M}$*, then so do all shortest paths on* $\mathbf{P}_n$ *between them.*

We are now equipped to prove Theorem 3. Note that we have demonstrated that semi-graphoids and convex rank tests can be regarded as sets of edges of $\mathbf{P}_n$, so we will show that their axiom systems are equivalent. We first show that a semi-graphoid satisfies (PC). Consider $\delta, \delta'$ in the same class $C$ of a semi-graphoid, and let $\delta'' \in \mathcal{L}(\delta, \delta')$. Further, let $p$ be a shortest path from $\delta$ to $\delta''$ (so, $p\delta = \delta''$), and let $q$ be a shortest path from $\delta''$ to $\delta'$. We claim

that $qp$ is a shortest path from $\delta$ to $\delta'$, and thus $\delta'' \in C$ by Lemma 7. Suppose $qp$ is not a shortest path. Then, we can obtain a shorter path in the semi-graphoid by some sequence of substitutions according to (BS), (BH), and (BN'). Only (BN') decreases the length of a path, so the sequence must involve (BN'). Therefore, there is some $i$, $j$ in $[n]$, such that their positions relative to each other are reversed twice in $qp$. But $p$ and $q$ are shortest paths, hence one reversal occurs in each $p$ and $q$. Then $\delta$ and $\delta'$ agree on whether $i > j$ or $j > i$, but the reverse holds in $\delta''$, contradicting $\delta'' \in \mathcal{L}(\delta, \delta')$. Thus every semi-graphoid is a pre-convex rank test.

Now, we show that a semi-graphoid corresponds to a fan. Consider the cone corresponding to a class $C$. We need only show that it meets any other cone in a shared face. Since $C$ is a cone of a coarsening of the $S_n$-fan, each nonmaximal face of $C$ lies in a hyperplane $H = \{x_i = x_j\}$. Suppose a face of $C$ coincides with the hyperplane $H$ and that $i > j$ in $C$. A vertex $\delta$ borders $H$ if $i$ and $j$ are adjacent in $\delta$. We will show that if $\delta, \delta' \in C$ border $H$, then their reflections $\hat{\delta} = \delta_1 \dots ji \dots \delta_n$ and $\hat{\delta'} = \delta'_1 \dots ji \dots \delta'_n$ both lie in some class $C'$. Consider a 'great circle' path between $\delta$ and $\delta'$ which stays closest to $H$: all vertices in the path have $i$ and $j$ separated by at most one position, and no two consecutive vertices have $i$ and $j$ nonadjacent. This is a shortest path, so it lies in $C$, by Lemma 7. Using the square and hexagon axioms (Observation 6), we see that the reflection of the path across $H$ is a path in the semi-graphoid that connects $\hat{\delta}$ to $\hat{\delta'}$ (Figure 3). Thus a semigraphoid is a convex rank test.

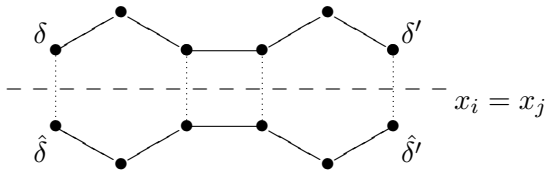



Figure 3: Reflecting a path across a hyperplane.

Finally, if $\mathbf{M}$ is a set of edges of $\mathbf{P}_n$, representing a convex rank test, then it is easy to show that $\mathbf{M}$ satisfies the square and hexagon axioms. This completes the proof of Theorem 3.

**Remark 8.** For $n = 3$ there are 40 pre-convex rank tests, but only 22 of them are convex rank tests. The corresponding CI models are shown in Figure 5.6 on page 108 in (Studený, 2005).

## 4 The submodular cone

In this section we examine a subclass of the convex rank tests. Let $2^{[n]}$ denote the collection of all subsets of $[n] = \{1, 2, \dots, n\}$. Any real-valued function $w : 2^{[n]} \to \mathbb{R}$ defines a convex polytope $Q_w$ of dimension $\leq n - 1$ as follows:

$$Q_w := \big\{ x \in \mathbb{R}^n : x_1 + x_2 + \cdots + x_n = w([n]) \text{ and } \textstyle\sum_{i \in I} x_i \leq w(I) \text{ for all } \emptyset \neq I \subseteq [n] \big\}.$$

A function $w : 2^{[n]} \to \mathbb{R}$ is called *submodular* if $w(I) + w(J) \geq w(I \cap J) + w(I \cup J)$ for $I, J \subseteq [n]$.

**Proposition 9.** *A function $w : 2^{[n]} \to \mathbb{R}$ is submodular if and only if the normal fan of the polyhedron $Q_w$ is a coarsening of the $S_n$-fan.*

This follows from greedy maximization as in (Lovász, 1983). Note that the function $w$ is submodular if and only if the optimal solution of

$$\text{maximize } u \cdot x \text{ subject to } x \in Q_w$$

depends only on the permutation equivalence class of $u$. Thus, solving this linear programming problem constitutes a convex rank test. Any such test is called a *submodular rank test.*

A convex polytope is a *(Minkowski) summand* of another polytope if the normal fan of the latter refines the normal fan of the former. The polytope $Q_w$ that represents a submodular rank test is a summand of the permutohedron $\mathbf{P}_n$.

**Theorem 10.** *The following combinatorial objects are equivalent for any positive integer $n$:*
1. *submodular rank tests,*
2. *summands of the permutohedron $\mathbf{P}_n$,*
3. *structural conditional independence models,*
4. *faces of the submodular cone $\mathbf{C}_n$ in $\mathbb{R}^{2^n}$.*

We have $1 \iff 2$ from Proposition 9, and $1 \iff 3$ follows from (Studený, 2005). Further $3 \iff 4$ holds by definition.

The *submodular cone* is the cone $\mathbf{C}_n$ of all submodular functions $w : 2^{[n]} \to \mathbb{R}$. Working modulo its lineality space $\mathbf{C}_n \cap (-\mathbf{C}_n)$, we regard $\mathbf{C}_n$ as a pointed cone of dimension $2^n - n - 1$.

**Remark 11.** All 22 convex rank tests for $n = 3$ are submodular. The submodular cone $\mathbf{C}_3$ is a 4-dimensional cone whose base is a bipyramid. The polytopes $Q_w$, as $w$ ranges over the faces of $\mathbf{C}_3$, are all the Minkowski summands of $\mathbf{P}_3$.

**Proposition 12.** *For $n \geq 4$, there exist convex rank tests that are not submodular rank tests. Equivalently, there are fans that coarsen the $S_n$-fan but are not the normal fan of any polytope.*

This result is stated in Section 2.2.4 of (Studený, 2005) in the following form: "There exist semi-graphoids that are not structural."

We answered Question 4.5 posed in (Postnikov, 2006) by finding a non-submodular convex rank test in which all the posets $P_i$ are trees:

$$\mathcal{M} \;=\; \{2 \perp\!\!\!\perp 3|\{1,4\},\, 1 \perp\!\!\!\perp 4|\{2,3\},\\ 1 \perp\!\!\!\perp 2|\emptyset,\, 3 \perp\!\!\!\perp 4|\emptyset\,\}.$$

**Remark 13.** For $n = 4$ there are 22108 submodular rank tests, one for each face of the 11-dimensional cone $\mathbf{C}_4$. The base of this submodular cone is a polytope with $f$-vector $(1, 37, 356, 1596, 3985, 5980, 5560, 3212, 1128, 228, 24, 1)$.

**Remark 14.** For $n = 5$ there are 117978 coarsest submodular rank tests, in 1319 symmetry classes. We confirmed this result of (Studený, 2000) with `POLYMAKE` (Gawrilow, 2000).

We now define a class of submodular rank tests, which we call *Minkowski sum of simplices (MSS) tests*. Note that each subset $K$ of $[n]$ defines a submodular function $w_K$ by setting $w_K(I) = 1$ if $K \cap I$ is non-empty and $w_K(I) = 0$ if $K \cap I$ is empty. The corresponding polytope $Q_{w_K}$ is the simplex $\Delta_K = \mathrm{conv}\{e_k : k \in K\}$.

Now consider an arbitrary subset $\mathcal{K} = \{K_1, K_2, \ldots, K_r\}$ of $2^{[n]}$. It defines the submodular function $w_\mathcal{K} = w_{K_1} + w_{K_2} + \cdots + w_{K_r}$. The corresponding polytope is the Minkowski sum

$$\Delta_\mathcal{K} \quad = \quad \Delta_{K_1} + \Delta_{K_2} + \cdots + \Delta_{K_r}.$$

The associated MSS test $\tau_\mathcal{K}$ is defined as follows. Given $\rho \in S_n$, we compute the number of indices $j \in [r]$ such that $\max\{\rho_k : k \in K_j\} = \rho_i$, for each $i \in [n]$. The signature $\tau_\mathcal{K}(\rho)$ is the vector in $\mathbb{N}^n$ whose $i$th coordinate is that number. Few submodular rank tests are MSS tests:

**Remark 15.** For $n = 3$, among the 22 submodular rank tests, only 15 are MSS tests. For $n = 4$, among the 22108, only 1218 are MSS.

## 5 Graphical tests

Graphical models are fundamental in statistics, and they also lead to a useful class of rank tests. First we show how to associate a semi-graphoid to a family $\mathcal{K}$. Let $\mathcal{F}_{w_\mathcal{K}}$ be the normal fan of $Q_{w_\mathcal{K}}$. We write $\mathcal{M}_\mathcal{K}$ for the CI model derived from $\mathcal{F}_{w_\mathcal{K}}$ using the bijection in Theorem 3.

**Proposition 16.** *The semi-graphoid $\mathcal{M}_\mathcal{K}$ is the set of CI statements $i \perp\!\!\!\perp j \,|\, K$ which satisfy the following property: all sets containing $\{i, j\}$ and contained in $\{i, j\} \cup [n]\backslash K$ are not in $\mathcal{K}$.*

Let $G$ be a graph with vertex set $[n]$. We define $\mathcal{K}(G)$ to be the collection of all subsets $K$ of $[n]$ such that the induced subgraph of $G|_K$ is connected. Recall that the *undirected graphical model* (or *Markov random field*) derived from the graph $G$ is the set $\mathcal{M}^G$ of CI statements:

$$\mathcal{M}^G \;=\; \{\, i \perp\!\!\!\perp j \,|\, C \;:\; \text{the restriction of } G \text{ to } [n]\backslash C \text{ contains no path from } i \text{ to } j\,\}.$$

The polytope $\Delta_G = \Delta_{\mathcal{K}(G)}$ is the *graph associahedron*, which is a well-studied object in combinatorics (Carr, 2004; Postnikov, 2005). The next theorem is derived from Proposition 16.

**Theorem 17.** *The CI model induced by the graph associahedron coincides with the graphical model $\mathcal{M}^G$, i.e., $\mathcal{M}_{\mathcal{K}(G)} = \mathcal{M}^G$.*

There is a natural involution $*$ on the set of all CI statements which is defined as follows:

$$(i \perp\!\!\!\perp j \,|\, C)^* \quad := \quad i \perp\!\!\!\perp j \,|\, [n]\backslash(C \cup \{i, j\}).$$

If $\mathcal{M}$ is any CI model, then the CI model $\mathcal{M}^*$ is obtained by applying the involution $*$ to all the CI statements in the model $\mathcal{M}$. Note that this involution was called duality in (Matúš, 1992).

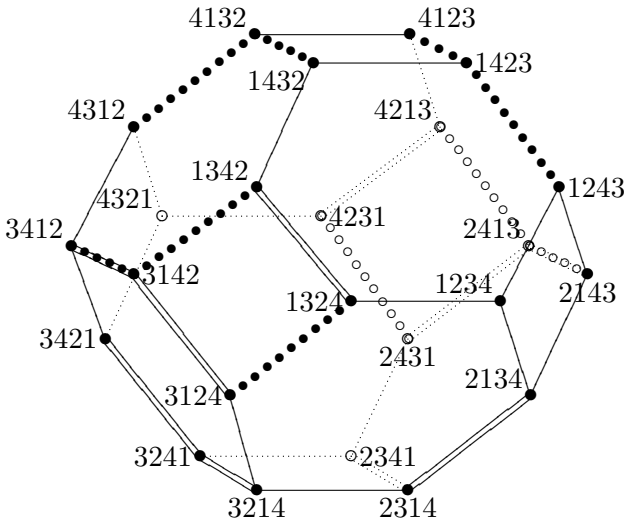



Figure 4: The permutohedron $\mathbf{P}_4$. Double edges indicate the test $\tau_{\mathcal{K}(G)}$ when $G$ is the path. Edges with large dots indicate the test $\tau^*_{\mathcal{K}(G)}$.

The *graphical tubing rank test* $\tau^*_{\mathcal{K}(G)}$ is the test associated with $\mathcal{M}^*_{\mathcal{K}(G)}$. It can be obtained by a construction similar to the MSS test $\tau_{\mathcal{K}}$, with the function $w_{\mathcal{K}}$ defined differently and supermodular. The *graphical model rank test* $\tau_{\mathcal{K}(G)}$ is the MSS test of the set family $\mathcal{K}(G)$.

We next relate $\tau_{\mathcal{K}(G)}$ and $\tau^*_{\mathcal{K}(G)}$ to a known combinatorial characterization of the graph associahedron $\Delta_G$. Two subsets $A, B \subset [n]$ are *compatible* for the graph $G$ if one of the following conditions holds: $A \subset B$, $B \subset A$, or $A \cap B = \emptyset$, and there is no edge between any node in $A$ and $B$. A *tubing* of the graph $G$ is a subset $\mathbf{T}$ of $2^{[n]}$ such that any two elements of $\mathbf{T}$ are compatible. Carr and Devadoss (2005) showed that $\Delta_G$ is a simple polytope whose faces are in bijection with the tubings.

**Theorem 18.** *The following four combinatorial objects are isomorphic for any graph $G$ on $[n]$:*
- *the graphical model rank test $\tau_{\mathcal{K}(G)}$,*
- *the graphical tubing rank test $\tau^*_{\mathcal{K}(G)}$,*
- *the fan of the graph associahedron $\Delta_G$,*
- *the simplicial complex of all tubings on $G$.*

The maximal tubings of $G$ correspond to vertices of the graph associahedron $\Delta_G$. When $G$ is the path of length $n$, then $\Delta_G$ is the *associahedron*, and when it is a cycle, $\Delta_G$ is the *cyclohedron*. The number of classes in the tubing test $\tau^*_{\mathcal{K}(G)}$ is the $G$-Catalan number of (Postnikov, 2005). This number is $\frac{1}{n+1}\binom{2n}{n}$ for the *associahedron test* and $\binom{2n-2}{n-1}$ for the *cyclohedron test*.

## 6 Enumerating linear extensions

In this paper we introduced a hierarchy of rank tests, ranging from pre-convex to graphical. Rank tests are applied to data vectors $u \in \mathbb{R}^n$, or permutations $\pi \in S_n$, and locate their cones. In order to determine the significance of a data vector, one needs to compute the quantity $|\,\tau^{-1}\big(\tau(\pi)\big)\,|$, and possibly the probabilities of other maximal cones. These cones are indexed by posets $P_1, P_2, \ldots, P_k$ on $[n]$, and the probability computations are equivalent to finding the cardinality of some of the sets $\mathcal{L}(P_i)$.

We now present our methods for computing linear extensions. If the rank test is a tubing test then this computation is done as follows. From the given permutation, we identify its signature (image under $\tau$), which we may assume is its $G$-tree $\mathbf{T}$ (Postnikov, 2005). Suppose the root of the tree $\mathbf{T}$ has $k$ children, each of which is a root of a subtree $\mathbf{T}^i$ for $i = 1, \ldots, k$. Writing $|\mathbf{T}^i|$ for the number of nodes in $\mathbf{T}^i$, we have

$$|\,\tau^{-1}(\mathbf{T})\,| = \binom{\sum_{i=1}^k |\mathbf{T}^i|}{|\mathbf{T}^1|, \ldots, |\mathbf{T}^k|} \left( \prod_{i=1}^k |\tau^{-1}(\mathbf{T}^i)| \right).$$

This recursive formula can be translated into an efficient iterative algorithm. In (Willbrand, 2005) the analogous problem is raised for the test in Example 3. A determinantal formula for (1) appears in (Stanley, 1997, page 69).

For an arbitrary convex rank test we proceed as follows. The test is specified (implicitly or explicitly) by a collection of posets $P_1, \ldots, P_k$ on $[n]$. From the given permutation, we first identify the unique poset $P_i$ of which that permutation is a linear extension. We next construct the *distributive lattice* $L(P_i)$ of all order ideals of $P_i$. Recall that an *order ideal* is a subset $O$ of $[n]$ such that if $l \in O$ and $(k, l) \in P_i$ then $k \in O$. The set of all order ideals is a lattice with meet and join operations given by set

intersection $O \cap O'$ and set union $O \cup O'$. Knowledge of this distributive lattice $L(P_i)$ solves our problem because the linear extensions of $P_i$ are precisely the maximal chains of $L(P_i)$. Computing the number of linear extensions is #P-complete (Brightwell, 1991). Therefore we developed efficient heuristics to build $L(P_i)$.

The key algorithmic task is the following: given a poset $P_i$ on $[n]$, compute an efficient representation of the distributive lattice $L(P_i)$. Our program for performing rank tests works as follows. The input is a permutation $\pi$ and a rank test $\tau$. The test $\tau$ can be specified either

- by a list of posets $P_1, \ldots, P_k$ (pre-convex),
- or by a semigraphoid $\mathcal{M}$ (convex rank test),
- or by a submodular function $w : 2^{[n]} \to \mathbb{R}$,
- or by a collection $\mathcal{K}$ of subsets of $[n]$ (MSS),
- or by a graph $G$ on $[n]$ (graphical test).

The output of our program has two parts. First, it gives the number $|\mathcal{L}(P_i)|$ of linear extensions, where the poset $P_i$ represents the equivalence class of $S_n$ specified by the data $\pi$. It also gives a representation of the distributive lattice $L(P_i)$, in a format that can be read by the **maple** package **posets** (Stembridge, 2004). Our software for the above rank tests is available at `www.bio.math.berkeley.edu/ranktests/`.

## Acknowledgments

This paper originated in discussions with Olivier Pourquié and Mary-Lee Dequéant in the DARPA Fundamental Laws of Biology Program, which supported Jason Morton, Lior Pachter, and Bernd Sturmfels. Anne Shiu was supported by a Lucent Technologies Bell Labs Graduate Research Fellowship. Oliver Wienand was supported by the Wipprecht foundation.

## References

G Brightwell and P Winkler. Counting linear extensions. *Order*, 8(3):225-242, 1991.

K Brown. *Buildings.* Springer, New York, 1989.

M Carr and S Devadoss. Coxeter complexes and graph associahedra. 2004. Available from `http://arxiv.org/abs/math.QA/0407229`.

E Gawrilow and M Joswig. Polymake: a framework for analyzing convex polytopes, in *Polytopes – Combinatorics and Computation*, eds. G Kalai and G M Ziegler, Birkhäuser, 2000, 43-74.

L Lovász. Submodular functions and convexity, in *Math Programming: The State of the Art*, eds. A Bachem, M Groetschel, and B Korte, Springer, 1983, 235-257.

F Matúš. Ascending and descending conditional independence relations, in *Proceedings of the Eleventh Prague Conference on Inform. Theory, Stat. Dec. Functions and Random Proc.*, Academia, B, 1992, 189-200.

F Matúš. Towards classification of semigraphoids. *Discrete Mathematics*, 277, 115-145, 2004.

EJG Pitman. Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society*, 4(1):119–130, 1937.

A Postnikov. Permutohedra, associahedra, and beyond. 2005. Available from `http://arxiv.org/abs/math/0507163`.

A Postnikov, V Reiner, L Williams. Faces of Simple Generalized Permutohedra. Preprint, 2006.

RP Stanley. *Enumerative Combinatorics* Volume I, Cambridge University Press, Cambridge, 1997.

J Stembridge. Maple packages for symmetric functions, posets, root systems, and finite Coxeter groups. Available from `www.math.lsa.umich.edu/~jrs/maple.html`.

M Studený. *Probablistic conditional independence structures.* Springer Series in Information Science and Statistics, Springer-Verlag, London, 2005.

M Studený, RR Bouckaert, and T Kocka. Extreme supermodular set functions over five variables. *Institute of Information Theory and Automation*, Research report n. 1977, Prague, 2000.

J Tits. *Le problème des mots dans les groupes de Coxeter.* Symposia Math., 1:175-185, 1968.

K Willbrand, F Radvanyi, JP Nadal, JP Thiery, and T Fink. Identifying genes from up-down properties of microarray expression series. *Bioinformatics*, 21(20):3859–3864, 2005.

G Ziegler. *Lectures on polytopes.* Vol. 152 of Graduate Texts in Mathematics. Springer-Verlag, 1995.

# An Empirical Study of Efficiency and Accuracy of Probabilistic Graphical Models

Jens Dalgaard Nielsen and Manfred Jaeger
Institut for Datalogi, Aalborg Universitet
Frederik Bajers Vej 7, 9220 Aalborg Ø, Denmark
{dalgaard,jaeger}@cs.aau.dk

**Abstract**

In this paper we compare Naïve Bayes (NB) models, general Bayes Net (BN) models and Probabilistic Decision Graph (PDG) models w.r.t. accuracy and efficiency. As the basis for our analysis we use graphs of size vs. likelihood that show the theoretical capabilities of the models. We also measure accuracy and efficiency empirically by running exact inference algorithms on randomly generated queries. Our analysis supports previous results by showing good accuracy for NB models compared to both BN and PDG models. However, our results also show that the advantage of the low complexity inference provided by NB models is not as significant as assessed in a previous study.

## 1 Introduction

Probabilistic graphical models (PGMs) have been applied extensively in machine learning and data mining research, and many studies have been dedicated to the development of algorithms for learning PGMs from data. Automatically learned PGMs are typically used for inference, and therefore efficiency and accuracy of the PGM w.r.t. inference are of interest when evaluating a learned model.

Among some of the most commonly used PGMs are the general Bayesian Network model (BN) and the Naïve Bayes model (NB). The BN model efficiently represents a joint probability distribution over a domain of discrete random variables by a factorization into independent local distributions. The NB model contains a number of *components* defined by an unobserved latent variable and models each discrete random variable as independent of all other variables within each component. Exact inference has linear time complexity in the size of the model when using NB models.

For the general BN model both exact and approximate inference are NP-hard (Cooper, 1987; Dagum and Luby, 1993).

Model-selection algorithms for learning PGMs typically use some conventional score-metric, searching for a model that optimises the metric. Penalised likelihood metrics like BIC, AIC and MDL are weighted sums of model accuracy and size. When the learned model is to be used for general inference, including a measure for inference complexity into the metric is relevant. Neither BIC, AIC nor MDL explicitly takes inference complexity into account when assessing a given model. Recently, several authors have independently emphasised the importance of considering inference complexity when applying learning in a real domain.

Beygelzimer and Rish (2003) investigate the tradeoff between model accuracy and efficiency. They only consider BN models for a given target distribution (in a learning setting, the target distribution is the empirical distribution defined by the data; more generally, the target distribution could be any distribution one wants to represent). For BNs *treewidth* is an adequate efficiency measure (defined as $k-1$, where $k$ is the size of the largest clique in an optimal junction tree). Tradeoff curves that plot treewidth against the best possible accuracy achievable with a given treewidth are introduced. These tradeoff curves can be used to investigate the *approximability* of a target distribution.

As an example for a distribution with poor approximability in this sense, Beygelzimer and Rish (2003) mention the parity distribution, which rep-

resents the parity function on $n$ binary inputs. An accurate representation of this distribution requires a BN of treewidth $n-1$, and any BN with a smaller treewidth can approximate the parity distribution only as well as the empty network.

The non-approximability of the parity distribution (and hence the impossibility of accurate models supporting efficient inference) only holds under the restriction to BN models with nodes corresponding exactly to the $n$ input bits. The use of other PGMs, or the use of latent variables in a BN representation, can still lead to accurate and computationally efficient representations of the parity distribution.

Motivated by some distribution's refusal to be efficiently approximated by BN models, the PGM language of probabilistic decision graph (PDG) models was developed (Jaeger, 2004). In particular, the parity distribution is representable by a PDG that has inference complexity linear in $n$. In a recent study an empirical analysis of the approximations offered by BN and PDG models learned from real-world data was conducted (Jaeger et al., 2006). Similar to the tradeoff curves of (Beygelzimer and Rish, 2003), Jaeger et al. (2006) used graphs showing likelihood of data vs. size of the model for the analysis of accuracy vs. complexity. The comparison of PDGs vs. BNs did not produce a clear winner, and the main lesson was that the models offer surprisingly similar tradeoffs when learned from real data.

Also motivated by considerations of model accuracy and efficiency, Lowd and Domingos (2005) in a recent study compared NB and BN models. NB models can potentially offer accuracy-efficiency tradeoff behaviors that for some distributions differ from those provided by standard BN representation (although NBs do not include the latent class models that allow an efficient representation of the parity distribution). Lowd and Domingos (2005) determine inference complexity empirically by measuring inference times on randomly generated queries. The inferences are computed exactly for NB models, but for BN models approximate methods were used (Gibbs sampling and loopy belief propagation). Lowd and Domingos (2005) conclude that NB models offer approximations that are as accurate as those offered by BN models, but in terms of inference complexity the NB models are reported to be orders of magnitude faster than BN models.

Our present paper extend these previous works in two ways. First, we conduct a comparative analysis of accuracy vs. efficiency tradeoffs for three type of PGMs: BN, NB and PDG models. Our results show that in spite of theoretical differences BN, NB and PDG models perform surprisingly similar when learned from real data, and no single model is consistently superior. Second, we investigate the theoretical and empirical efficiency of exact inference for all models. This analysis somewhat differs from the analysis in (Lowd and Domingos, 2005), where only approximate inference was considered for BNs. The latter approach can lead to somewhat unfavorable results for BNs, because approximate inference can be much slower than exact inference for models still amenable to exact inference. Our results show that while NB models are still very competitive w.r.t. accuracy, exact inference in BN models is often tractable and differences in empirically measured run-times are typically not significant.

## 2 Probabilistic Graphical Models

In this section we introduce the three types of models that we will use in our experiments; the general Bayesian Network (BN), the Naïve Bayesian Network (NB) and the Probabilistic Decision Graph (PDG).

### 2.1 Bayesian Network Models

BNs (Jensen, 2001; Pearl, 1988) are a class of probabilistic graphical models that represent a joint probability distribution over a domain $\mathbf{X}$ of discrete random variables through a factorization of independent local distributions or factors. The structure of a BN is a directed acyclic graph (DAG) $G = (\mathbf{V}, \mathbf{E})$ of nodes $\mathbf{V}$ and directed edges $\mathbf{E}$. Each random variable $X_i \in \mathbf{X}$ is represented by a node $V_i \in \mathbf{V}$, and the factorization $\prod_{X_i \in \mathbf{X}} P(X_i|pa_G(X_i))$ (where $pa_G(X_i)$ is the set of random variables represented by parents of node $V_i$ in DAG $G$) defines the full joint probability distribution $P(\mathbf{X})$ represented by the BN model. By *size* of a BN we understand the size of the *representation*, i.e. the number of independent parameters. Exact inference is usually performed by first constructing a junction tree from the BN. Inference is then solvable in time linear in the size of the junction tree (Lauritzen and

Spiegelhalter, 1988), which may be exponential in the size of the BN from which it was constructed.

### 2.2 Naïve Bayes Models

The NB model represents a joint probability distribution over a domain $\mathbf{X}$ of discrete random variables by introducing an unobserved, latent variable $C$. Each state of $C$ is referred to as a *component*, and conditioned on $C$, each variable $X_i \in \mathbf{X}$ is assumed to be independent of all other variables in $\mathbf{X}$. This yields the simple factorization: $P(\mathbf{X}, C) = P(C)\prod_{X_i \in \mathbf{X}} P(X_i|C)$. Exact inference is computable in time linear in the representation size of the NB model.

### 2.3 Probabilistic Decision Graph Models

PDGs are a fairly new language for probabilistic graphical modeling (Jaeger, 2004; Bozga and Maler, 1999). As BNs and NBs, PDGs represent a joint probability distribution over a domain of discrete random variables $\mathbf{X}$ through a factorization of local distributions. However, the structure of the factorization defined by a PDG is not based on a variable level independence model but on a certain kind of context specific independencies among the variables. A PDG can be seen as a two-layer structure, 1) a forest of tree-structures over all members of $\mathbf{X}$, and 2) a set of rooted DAG structures over *parameter* nodes, each holding a local distribution over one random variable. Figure 1(a) shows a forest $F$ of tree-structures over binary variables $\mathbf{X} = \{X_0, X_1 \ldots, X_5\}$, and figure 1(b) shows an example of a PDG structure based on $F$. For a complete semantics of the PDG model and algorithms for exact inference with linear complexity in the size of the model, the reader is referred to (Jaeger, 2004).

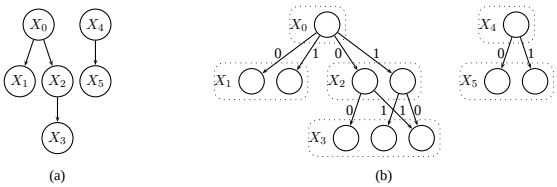



Figure 1: Example PDG. Subfigure (a) shows the a forest-structure $F$ over binary 5 variables, and (b) shows a full PDG structure based on $F$.

## 3 Elements of the Analysis

The goal of our analysis is to investigate the quality of PGMs learned from real data w.r.t. accuracy and inference efficiency. The appropriate notion of accuracy depends on the intended tasks for the model. Following (Jaeger et al., 2006; Lowd and Domingos, 2005; Beygelzimer and Rish, 2003) we use log-likelihood of the model given the data ($L(M, D)$) as a "global" measure of accuracy. Log-likelihood score is essentially equivalent to cross-entropy ($CE$) between the empirical distribution $P^D$ and the distribution $P^M$ represented in model $M$:

$$CE(P^D, P^M) = -H(P^D) - \frac{1}{|D|}L(M, D), \quad (1)$$

where $H(\cdot)$ is the entropy function. Observe that when $CE(P^D, P^M) = 0$ (when $P^D$ and $P^M$ are equal), then $L(M, D) = -|D| \cdot H(P^D)$. Thus, data entropy is an upper bound on the log-likelihood.

### 3.1 Theoretical Complexity vs. Accuracy

We use SL-curves (Jaeger et al., 2006) in our analysis of the theoretical performance of each PGM language. SL-curves are plots of size vs. likelihood. The size of model here is the *effective size*, i.e. a model complexity parameter, such that inference has linear time complexity in this parameter. For NB and PDG models this is the size of the model itself. For BN models it is the size of the junction tree constructed for inference.

### 3.2 Empirical Complexity and Accuracy

The size measure used in the SL-curves described in section 3.1 measures inference complexity only up to a linear factor. Following Lowd and Domingos (2005), we estimate the complexity of exact inference also empirically by measuring execution times for random queries. A query for model $M$ is solved by computing a conditional probability $P^M(\mathbf{Q} = \mathbf{q}|\mathbf{E} = \mathbf{e})$, where $\mathbf{Q}, \mathbf{E}$ are disjoint subsets of $\mathbf{X}$, and $\mathbf{q}, \mathbf{e}$ are instantiations of $\mathbf{Q}$, respectively $\mathbf{E}$. Queries are randomly generated as follows: first a random pair $\langle \mathbf{Q}_i, \mathbf{E}_i \rangle$ of subsets of variables is drawn from $\mathbf{X}$. Then, an instance $d_i$ is randomly drawn from the test data. The random query then is $P^M(\mathbf{Q} = d_i[\mathbf{Q}_i]|\mathbf{E} = d_i[\mathbf{E}_i])$, where

$d_i[\mathbf{Q}_i], d_i[\mathbf{E}_i]$ are the instantiations of $\mathbf{Q}$, respectively $\mathbf{E}$ in $d_i$. The empirical complexity is simply the average execution time for random queries. The empirical accuracy is measured by averaging $log(P^M(\mathbf{Q} = d_i[\mathbf{Q}_i]|\mathbf{E} = d_i[\mathbf{E}_i]))$ over the random queries. Compared to the global accuracy measure $L(M, D)$, this can be understood as a measure for "local" accuracy, i.e. restricted to specific conditional and marginal distributions of $P^M$.

## 4 Learning

In this section we briefly describe the algorithms we use for learning each type of PGM from data. For our analysis we need to learn a range of models with different efficiency vs. accuracy tradeoffs. For *score based* learning a general $\lambda$-score will be used (Jaeger et al., 2006):

$$S_\lambda(M, D) = \lambda \cdot L(M, D) - (1 - \lambda)|M|, \quad (2)$$

where $0 < \lambda < 1$, and $|M|$ is the size of model $M$. Equation (2) is a general score metric, as it becomes equivalent to common metrics as BIC, AIC and MDL for specific settings of $\lambda$[1]. By optimizing scores with different settings of $\lambda$ we get a range of models offering different tradeoffs between size and accuracy. Suitable ranges of $\lambda$-values were determined experimentally for each type of model using score score-based learning (BNs and PDGs).

### 4.1 Learning Bayesian Networks

We use the KES algorithm for learning BN models (Nielsen et al., 2003). KES performs model-selection in the space of equivalence classes of BN structures using a semi-greedy heuristic. A parameter $k \in [0 \dots 1]$ controls the level of greediness, where a setting of 0 is maximally stochastic and 1 is maximally greedy.

The $\lambda$-score used in the KES algorithm uses the size of the BN as the size parameter $|M|$, not the size of its junction tree. Clearly, it would be desirable to score BNs directly by the size of their junction trees, but this appears computationally infeasible. Thus, our SL-curves for BNs do not show for a given accuracy level the smallest possible size of a junction tree achieving that accuracy, but the size of a junction tree we were able to find using existing state-of-the-art learning and triangulation methods.

### 4.2 Learning Naïve Bayes Net Models

For learning NB models we have implemented a version of the NBE algorithm (Lowd and Domingos, 2005) for learning NB models. As the structure of NB models is fixed, the task reduces to learning the number of states in the latent variable $C$, and the parameters of the model. Learning in the presence of the latent components is done by standard Expectation Maximization (EM) approach, following (Lowd and Domingos, 2005; Karciauskas et al., 2004). Learning a range of models is done by incrementally increasing the number of states of $C$, and outputting the model learned for each cardinality. In this way we obtain a range of models that offer different complexity vs. accuracy tradeoffs. Note that no structure-score like (2) is required as the structure is fixed.

### 4.3 Learning Probabilistic Decision Graphs

Learning of PDGs is done using the model-selection algorithm presented in (Jaeger et al., 2006). Using local transformations as search operators, the algorithm performs a search for a structure that optimises $\lambda$-score.

## 5 Experiments

We have produced SL-curves and empirically measured inference times and accuracy, on 5 different datasets (see table 1) from the UCI repository[2]. These five datasets are a representative sample of the 50 datasets used in the extensive study by Lowd and Domingos (2005).

We used the same versions of the datasets as used by Lowd and Domingos (2005). Specifically, the partitioning into training (90%) and test (10%) sets was the same, continuous variables were discretized into five equal frequency bins, and missing values were interpreted as special states of the variables.

For measuring the empirical efficiency and accuracy, we generated random queries as described in 3.2 consisting of 1 to 5 query variables $\mathbf{Q}$ and 0 to 5 evidence variables $\mathbf{E}$.

[1] E.g. (2) with $\lambda = \frac{log|D|}{2+\log|D|}$ corresponds to BIC

[2] `http://www.ics.uci.edu/~mlearn`

Table 1: Datasets used for experiments.

| Dataset | #Vars | training | test |
|---|---|---|---|
| Poisonous Mushroom | 23 | 7337 | 787 |
| King, Rook vs. King | 7 | 25188 | 2868 |
| Pageblocks | 11 | 4482 | 574 |
| Abalone | 9 | 3758 | 419 |
| Image Segmentation | 17 | 2047 | 263 |

For inference in BN and NB models we used the junction-tree algorithm implemented in the inference engine in Hugin[3] through the Hugin Java API. For inference in PDGs, the method described in (Jaeger, 2004) was implemented in Java. BN and NB experiments were performed on a standard laptop, 1.6GHz Pentium CPU with 512Mb RAM running Linux. PDG experiments were performed on a Sun Fire280R, 900Mhz SPARC CPU with 4Gb RAM running Solaris 9.

## 6 Results

Table 2 shows SL-curves for each dataset and PGM, both for training (left column) and test sets (right column).

Lowd and Domingos (2005) based their comparison on single BN and NB models. The NB models were selected by maximizing likelihood on a hold-out set. Crosses $\times$ in the right column of table 2 indicate the size-likelihood values obtained by the NB models reported in (Lowd and Domingos, 2005).

The first observation we make from table 2 is that no single model language consistently dominates the others. The plots in the left column shows that BN models have the lowest log-likelihood measured on training data consistently for models larger than some small threshold. For Abalone and Image Segmentation, this characteristic is mitigated in the plots for the test-data, where especially PDGs seem to overfit the training-data and accordingly receives low log-likelihood score on the test-data.

The overall picture in table 2 is that in terms of accuracy, BNs and NBs are often quite similar (King, Rook vs. King is the only exception). This is consistent with what Lowd and Domingos (2005) have found. However, Lowd and Domingos (2005) reported big differences in inference complexity when comparing exact inference in NB to approximate methods in BNs. We do not observe this tendency when considering exact inference for both BNs and NBs. Our results show that we can learn BNs that are within reach of exact inference methods, and that the theoretical inference complexity as measured by effective model size mostly is similar for a given accuracy level for all three PGM languages.

Effective model size measures actual inference time only up to a linear factor. In order to determine whether there possibly are huge (orders of magnitude) differences in these linear factors, we measure the actual inference time on our random queries. The left column of table 3 shows the average inference time for 1000 random queries with 4 query and 3 evidence variables (results for other numbers of query and evidence variables were very similar). We observe that this empirical complexity behaves almost indistinguishably for BN and NB models. This is not surprising, since both models use the Hugin inference engine [4]. The results do show, however, that the different structures of the junction trees for BN and NB models do not have a significant impact on runtime. The linear factor for PDG inference in these experiments is about 4 times larger than that for BN/NB inference[5]. Seeing that we use a proof-of-concept prototype Java implementation for PDGs, and the commercial Hugin inference engine for BNs and NBs, this indicates that PDGs are competitive in practice, not only according to theoretical complexity analyses.

The right column in table 3 shows the empirical (local) accuracy obtained for 1000 random queries with 4 query and 3 evidence variables. Overall, the results are consistent with the global accuracy on the test data (table 2, right column). The differences observed for the different PGMs in table 2 can also be seen in table 3, though the discrepancies tend

[3] `http://www.hugin.com`

[4] Zhang (1998) shows that variable elimination can be more efficient than junction tree-based inference. However, his results do not indicate that we would obtain substantially different results if we used variable elimination in our experiments.

[5] This factor has to be viewed with caution, since PDG inference was run on a machine with a slower CPU but more main memory. When running PDG inference on the same machine as NB/BN inference, we observed overall a similar performance, but more deviations from a strictly linear behavior (in table 3 still visible to some degree for the Mushroom and Abalone data). These deviations seem mostly attributable to the memory management in the Java runtime environment.

Table 2: SL-curves for train-sets (left column) and test-sets (right column). The crosses $\times$ marks the NB models reported by Lowd and Domingos (2005). $-H(D)$ (minus data-entropy) is plotted as a horizontal line.

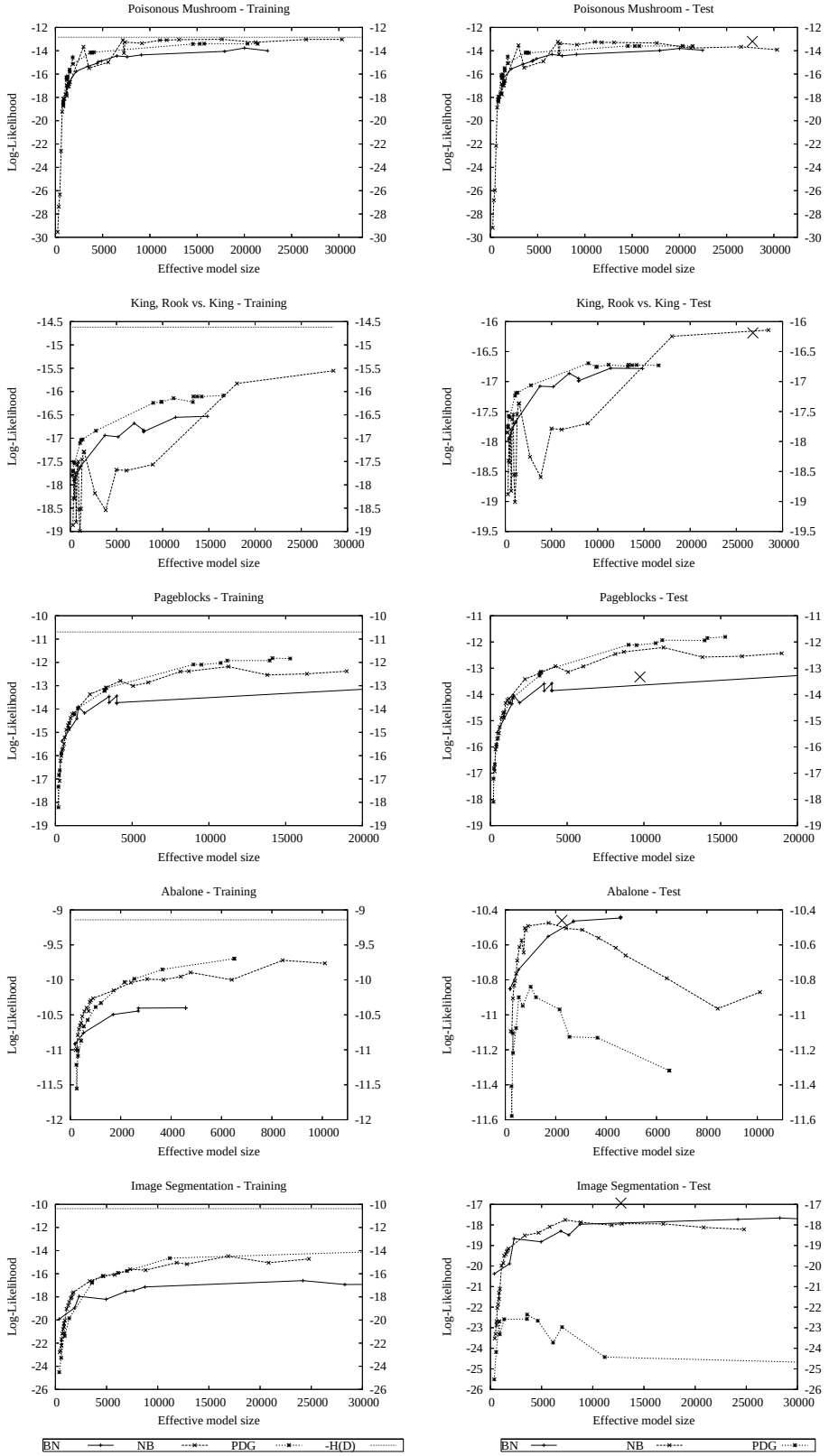
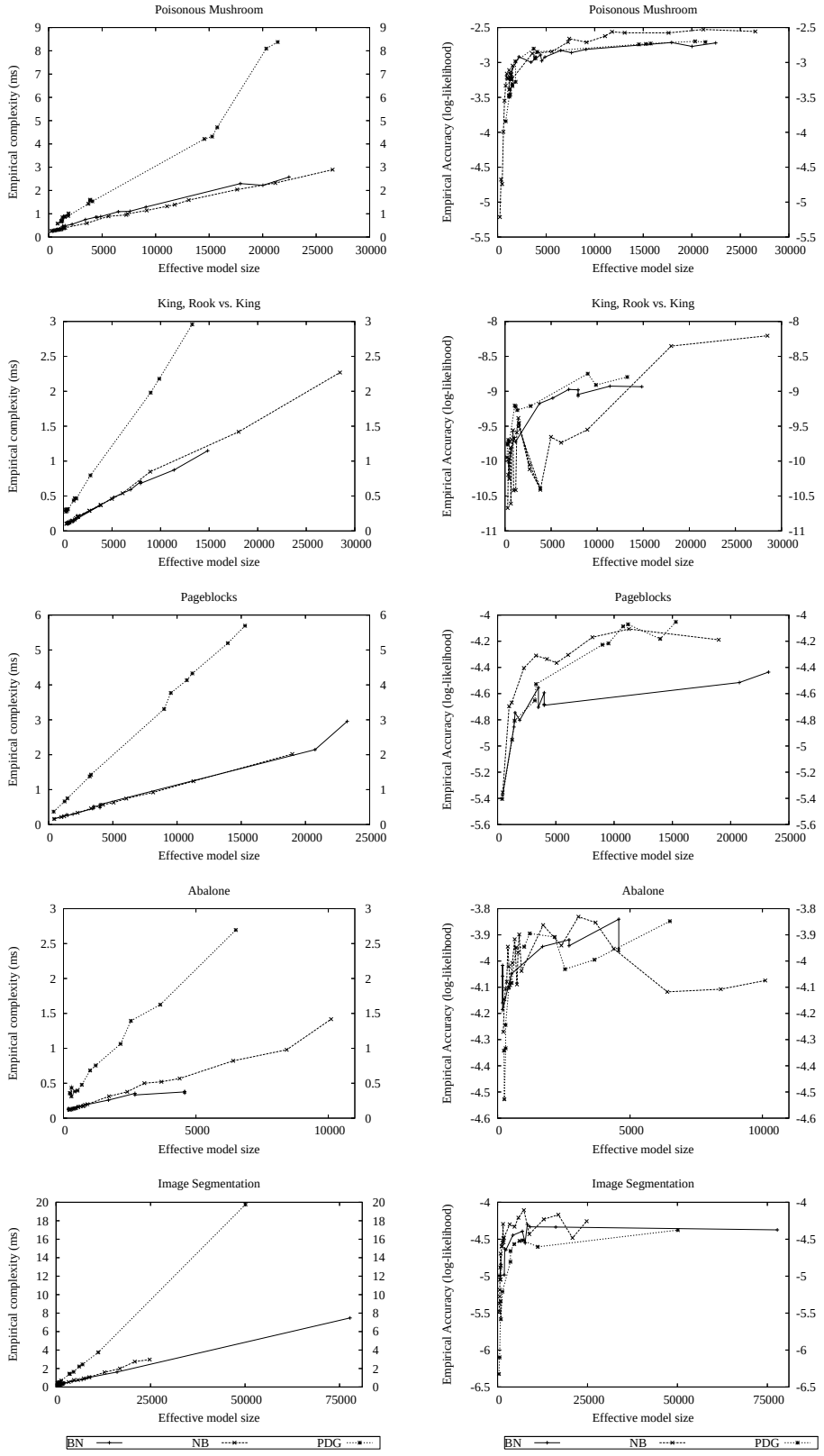


Table 3: Empirical efficiency (left column) and accuracy (right column) for 4 query and 3 evidence variables.



to become less pronounced on the random queries as on global likelihood (particularly for PDGs in the image segmentation data). One possible explanation for this is that low global likelihood scores are mostly due to a few test cases whose joint instantiation of the variables are given low probability by a model, and that these isolated low-probability configurations are seldom met with in the random queries.

## 7 Conclusion

Motivated by several previous, independent studies on the tradeoff between model accuracy and efficiency in different PGM languages, we have investigated the performance of BN, NB, and PDG models. Our main findings are: 1) In contrast to potentially widely different performance on artificial examples (e.g. the parity distribution), we observe a relatively uniform behavior of all three languages on real-life data. 2) Our results confirm the conclusions of Lowd and Domingos (2005) that the NB model is a viable alternative to the BN model for general purpose probabilistic modeling and inference. However, the order-of-magnitude advantages in inference complexity could not be confirmed when comparing exact inference methods for both types of models. 3) Previous theoretical complexity analyses for inference in PDG models now have been complemented with empirical results showing also the practical competitiveness of PDGs.

## Acknowledgment

We thank Daniel Lowd for providing the preprocessed datasets we used in our experiments. We thank the AutonLab at CMU for making their efficient software libraries available to us.

## References

Beygelzimer, A. and Rish, I.: 2003, Approximability of probability distributions, *Advances in Neural Information Processing Systems 16*, The MIT Press.

Bozga, M. and Maler, O.: 1999, On the representation of probabilities over structured domains, *Proceedings of the 11th International Conference on Computer Aided Verification*, Springer, pp. 261–273.

Cooper, G. F.: 1987, Probabilistic inference using belief networks is NP-hard, *Technical report*, Knowledge Systems Laboratory, Stanford University.

Dagum, P. and Luby, M.: 1993, Approximating probabilistic inference in Bayesian belief networks is NP-hard, *Artificial Intelligence* **60**, 141–153.

Jaeger, M.: 2004, Probabilistic decision graphs - combining verification and ai techniques for probabilistic inference, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **12**, 19–42.

Jaeger, M., Nielsen, J. D. and Silander, T.: 2006, Learning probabilistic decision graphs, *International Journal of Approximate Reasoning* **42**(1-2), 84–100.

Jensen, F. V.: 2001, *Bayesian Networks and Decision Graphs*, Springer.

Karciauskas, G., Kočka, T., Jensen, F. V., Larrañaga, P. and Lozano, J. A.: 2004, Learning of latent class models by splitting and merging components, *Proceedings of the Second European Workshop on Probabilistic Graphical Models*, pp. 137–144.

Lauritzen, S. L. and Spiegelhalter, D. J.: 1988, Local computations with probabilities on graphical structures and their application to expert systems, *Journal of the Royal Statistical Society* **50**(2), 157–224.

Lowd, D. and Domingos, P.: 2005, Naive Bayes models for probability estimation, *Proceedings of the Twentysecond International Conference on Machine Learning*, pp. 529–536.

Nielsen, J. D., Kočka, T. and Peña, J. M.: 2003, On local optima in learning Bayesian networks, *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers, pp. 435–442.

Pearl, J.: 1988, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann Publishers.

Zhang, N. L.: 1998, Computational properties of two exact algorithms for Bayesian networks, *Applied Intelligence* **9**, 173–183.

# Adapting Bayes Network Structures to Non-stationary Domains

Søren Holbech Nielsen and Thomas D. Nielsen
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7E
9220 Aalborg Ø, Denmark

**Abstract**

When an incremental structural learning method gradually modifies a Bayesian network (BN) structure to fit observations, as they are read from a database, we call the process structural adaptation. Structural adaptation is useful when the learner is set to work in an unknown environment, where a BN is to be gradually constructed as observations of the environment are made. Existing algorithms for incremental learning assume that the samples in the database have been drawn from a single underlying distribution. In this paper we relax this assumption, so that the underlying distribution can change during the sampling of the database. The method that we present can thus be used in unknown environments, where it is not even known whether the dynamics of the environment are stable. We briefly state formal correctness results for our method, and demonstrate its feasibility experimentally.

## 1 Introduction

Ever since Pearl (1988) published his seminal book on Bayesian networks (BNs), the formalism has become a widespread tool for representing, eliciting, and discovering probabilistic relationships for problem domains defined over discrete variables. One area of research that has seen much activity is the area of learning the structure of BNs, where probabilistic relationships for variables are discovered, or inferred, from a database of observations of these variables (main papers in learning include (Heckerman, 1998)). One part of this research area focuses on incremental structural learning, where observations are received sequentially, and a BN structure is gradually constructed along the way without keeping all observations in memory. A special case of incremental structural learning is structural adaptation, where the incremental algorithm maintains one or more candidate structures and applies changes to these structures as observations are received. This particular area of research has received very little attention, with the only results that we are aware of being (Buntine, 1991; Lam and Bacchus, 1994; Lam, 1998; Friedman and Goldszmidt, 1997; Roure, 2004).

These papers all assume that the database of observations has been produced by a stationary stochastic process. That is, the ordering of the observations in the database is inconsequential. However, many real life observable processes cannot really be said to be invariant with respect to time: Mechanical mechanisms may suddenly fail, for instance, and non-observable effects may change abruptly. When human decision makers are somehow involved in the data generating process, these are almost surely not fully describable by the observables and may change their behaviour instantaneously. A simple example of a situation, where it is unrealistic to expect a stationary generating process, is an industrial system, where some component is exchanged for one of another make. Similarly, if the coach of a soccer team changes the strategy of the team during a match, data on the play from after the chance would be distributed differently from that representing the time before.

In this work we relax the assumption on stationary data, opting instead for learning from data which is only "approximately" stationary. More concretely, we assume that the data generating process is piecewise stationary, as in the examples given above, and thus do not try to deal with data

where the data generating process changes gradually, as can happen when machinery is slowly being worn down.[1] Furthermore, we focus on domains where the shifts in distribution from one stationary period to the next is of a local nature (i.e. only a subset of the probabilistic relationships among variables change as the shifts take place).

## 2 Preliminaries

As a general notational rule we use bold font to denote sets and vectors ($\boldsymbol{V}$, $\boldsymbol{c}$, etc.) and calligraphic font to denote mathematical structures and compositions ($\mathcal{B}$, $\mathcal{G}$, etc.). Moreover, we shall use upper case letters to denote random variables or sets of random variables ($X$, $Y$, $\boldsymbol{V}$, etc.), and lower case letters to denote specific states of these variables ($x_4$, $y'$, $\boldsymbol{c}$, etc.). $\equiv$ is used to denote "defined as".

A BN $\mathcal{B} \equiv (\mathcal{G}, \boldsymbol{\Phi})$ over a set of discrete variables $\boldsymbol{V}$ consists of an acyclic directed graph (traditionally abbreviated DAG) $\mathcal{G}$, whose nodes are the variables in $\boldsymbol{V}$, and a set of conditional probability distributions $\boldsymbol{\Phi}$ (which we abbreviate CPTs for "conditional probability table"). A correspondence between $\mathcal{G}$ and $\boldsymbol{\Phi}$ is enforced by requiring that $\boldsymbol{\Phi}$ consists of one CPT $P(X|\mathbf{PA}_{\mathcal{G}}(X))$ for each variable $X$, specifying a conditional probability distribution for $X$ given each possible instantiation of the parents $\mathbf{PA}_{\mathcal{G}}(X)$ of $X$ in $\mathcal{G}$. A unique joint distribution $P_{\mathcal{B}}$ over $\boldsymbol{V}$ is obtained by taking the product of all the CPTs in $\boldsymbol{\Phi}$. When it introduces no ambiguity, we shall sometimes treat $\mathcal{B}$ as synonymous with its graph $\mathcal{G}$.

Due to the construction of $P_{\mathcal{B}}$ we are guaranteed that all dependencies inherent in $P_{\mathcal{B}}$ can be read directly from $\mathcal{G}$ by use of the *d-separation criterion* (Pearl, 1988). The d-separation criterion states that, if $X$ and $Y$ are d-separated by $\boldsymbol{Z}$, then it holds that $X$ is conditionally independent of $Y$ given $\boldsymbol{Z}$ in $P_{\mathcal{B}}$, or equivalently, if $X$ is conditionally dependent of $Y$ given $\boldsymbol{Z}$ in $P_{\mathcal{B}}$, then $X$ and $Y$ are not d-separated by $\boldsymbol{Z}$ in $\mathcal{G}$. In the remainder of the text, we use $X \perp\!\!\!\perp_{\mathcal{G}} Y \mid \boldsymbol{Z}$ to denote that $X$ is d-separated from $Y$ by $\boldsymbol{Z}$ in the DAG $\mathcal{G}$, and $X \perp\!\!\!\perp_{P} Y \mid \boldsymbol{Z}$ to denote that $X$ is conditionally independent of $Y$ given $\boldsymbol{Z}$ in the distribution $P$. The d-separation criterion is thus

$$X \perp\!\!\!\perp_{\mathcal{G}} Y \mid \boldsymbol{Z} \Rightarrow X \perp\!\!\!\perp_{P_{\mathcal{B}}} Y \mid \boldsymbol{Z},$$

for any BN $\mathcal{B} \equiv (\mathcal{G}, \boldsymbol{\Phi})$. The set of all conditional independence statements that may be read from a graph in this manner, is referred to as that graph's *d-separation properties*.

We refer to any two graphs over the same variables as being *equivalent* if they have the same d-separation properties. Equivalence is obviously an equivalence relation. Verma and Pearl (1990) proved that equivalent graphs necessarily have the same skeleton and the same v-structures.[2] The equivalence class of graphs containing a specific graph $\mathcal{G}$ can then be uniquely represented by the partially directed graph $\mathcal{G}^*$ obtained from the skeleton of $\mathcal{G}$ by directing links that participate in a v-structure in $\mathcal{G}$ in the direction dictated by $\mathcal{G}$. $\mathcal{G}^*$ is called the *pattern* of $\mathcal{G}$. Any graph $\mathcal{G}'$, which is obtained from $\mathcal{G}^*$ by directing the remaining undirected links, without creating a directed cycle or a new v-structure, is then equivalent to $\mathcal{G}$. We say that $\mathcal{G}'$ is a *consistent extension* of $\mathcal{G}^*$. The partially directed graph $\mathcal{G}^{**}$ obtained from $\mathcal{G}^*$, by directing undirected links as they appear in $\mathcal{G}$, whenever all consistent extensions of $\mathcal{G}^*$ agree on this direction, is called the *completed* pattern of $\mathcal{G}$. $\mathcal{G}^{**}$ is obviously a unique representation of $\mathcal{G}$'s equivalence class as well.

Given any joint distribution $P$ over $\boldsymbol{V}$ it is possible to construct a BN $\mathcal{B}$ such that $P = P_{\mathcal{B}}$ (Pearl, 1988). A distribution $P$ for which there is a BN $\mathcal{B}_P \equiv (\mathcal{G}_P, \boldsymbol{\Phi}_P)$ such that $P_{\mathcal{B}_P} = P$ and also

$$X \perp\!\!\!\perp_{P} Y \mid \boldsymbol{Z} \Rightarrow X \perp\!\!\!\perp_{\mathcal{G}_P} Y \mid \boldsymbol{Z}$$

holds, is called *DAG faithful*, and $\mathcal{B}_P$ (and sometimes just $\mathcal{G}_P$) is called a *perfect map*. DAG faithful distributions are important since, if a data generating process is known to be DAG faithful, then a perfect map can, in principle, be inferred from the data under the assumption that the data is representative of the distribution.

For any probability distribution $P$ over variables $\boldsymbol{V}$ and variable $X \in \boldsymbol{V}$, we define a *Markov boundary* (Pearl, 1988) of $X$ to be a set $\boldsymbol{S} \subseteq \boldsymbol{V} \setminus \{X\}$

[1] The changes in distribution of such data is of a continous nature, and adaptation of networks would probably be better accomplished by adjusting parameters in the net, rather than the structure itself.

[2] A triple of nodes $(X, Y, Z)$ constitutes a *v-structure* iff $X$ and $Z$ are non-adjacent and both are parents of $Y$.

such that $X \perp\!\!\!\perp_P \boldsymbol{V} \setminus (\boldsymbol{S} \cup \{X\}) \mid \boldsymbol{S}$ and this holds for no proper subset of $\boldsymbol{S}$. It is easy to see that if $P$ is DAG faithful, the Markov boundary of $X$ is uniquely defined, and consists of $X$'s parents, children, and children's parents in a perfect map of $P$. In the case of $P$ being DAG faithful, we denote the Markov boundary by $\mathbf{MB}_P(X)$.

## 3 The Adaptation Problem

Before presenting our method for structural adaptation, we describe the problem more precisely:

We say that a sequence is *samples from a piecewise DAG faithful distribution*, if the sequence can be partitioned into sets such that each set is a database sampled from a single DAG faithful distribution, and the *rank* of the sequence is the size of the smallest such partition. Formally, let $\mathcal{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_l)$ be a sequence of observations over variables $\boldsymbol{V}$. We say that $\mathcal{D}$ is sampled from a piecewise DAG faithful distribution (or simply that it is a piecewise DAG faithful sequence), if there are indices $1 = i_1 < \cdots < i_m = l + 1$, such that each of $\mathcal{D}_j = (\boldsymbol{d}_{i_j}, \ldots, \boldsymbol{d}_{i_{j+1}-1})$, for $1 \leq j \leq m - 1$, is a sequence of samples from a DAG faithful distribution. The rank of the sequence is defined as $\min_j i_{j+1} - i_j$, and we say that $m - 1$ is its *size* and $l$ its *length*. A pair of consecutive samples, $\boldsymbol{d}_i$ and $\boldsymbol{d}_{i+1}$, constitute a *shift* in $\mathcal{D}$, if there is $j$ such that $\boldsymbol{d}_i$ is in $\mathcal{D}_j$ and $\boldsymbol{d}_{i+1}$ is in $\mathcal{D}_{j+1}$. Obviously, we can have any sequence of observations being indistinguishable from a piecewise DAG faithful sequence, by selecting the partitions small enough, so we restrict our attention to sequences that are piecewise DAG faithful of at least rank $r$. However, we do not assume that neither the actual rank nor size of the sequences are known, and specifically we do not assume that the indices $i_1, \ldots, i_m$ are known.

The learning task that we address in this paper consists of incrementally learning a BN, while receiving a piecewise DAG faithful sequence of samples, and making sure that after each sample point the BN structure is as close as possible to the distribution that generated this point. Throughout the paper we assume that each sample is complete, so that no observations in the sequence have missing values. Formally, let $\mathcal{D}$ be a complete piecewise DAG faithful sample sequence of length $l$, and let $P_t$ be the distribution generating sample point $t$. Furthermore, let $\mathcal{B}_1, \ldots, \mathcal{B}_l$ be the BNs found by a structural adaptation method $M$, when receiving $\mathcal{D}$. Given a distance measure *dist* on BNs, we define the *deviance* of $M$ on $\mathcal{D}$ wrt. *dist* as

$$\mathit{dev}(M, \mathcal{D}) \equiv \frac{1}{l} \sum_{i=1}^{l} \mathit{dist}(\mathcal{B}_{P_i}, \mathcal{B}_i).$$

For a method $M$ to *adapt* to a DAG faithful sample sequence $\mathcal{D}$ wrt. *dist* then means that $M$ seeks to minimize its deviance on $\mathcal{D}$ wrt. *dist* as possible.

## 4 A Structural Adaptation Method

The method proposed here continuously monitors the data stream $\mathcal{D}$ and evaluates whether the last, say $k$, observations fit the current model. When this turns out not to be the case, we conclude that a shift in $\mathcal{D}$ took place $k$ observations ago. To adapt to the change, an immediate approach could be to learn a new network from the last $k$ cases. By following this approach, however, we will unfortunately loose all the knowledge gained from cases before the last $k$ observations. This is a problem if some parts of the perfect maps, of the two distributions on each side of the shift, are the same, since in such situations we re-learn those parts from the new data, even though they have not changed. Not only is this a waste of computational effort, but it can also be the case that the last $k$ observations, while not directly contradicting these parts, do not enforce them either, and consequently they are altered erroneously. Instead, we try to detect where in the perfect maps of the two distributions changes have taken place, and only learn these parts of the new perfect map. This presents challenges, not only in detection, but also in learning the changed parts and having them fit the non-changed parts seamlessly. Hence, the method consists of two main mechanisms: One, monitoring the current BN while receiving observations and detecting when and where the model should be changed, and two, re-learning the parts of the model that conflicts with the observations, and integrating the re-learned parts with the remaining parts of the model. These two mechanisms are described below in Sections 4.1 and 4.2, respectively.

### 4.1 Detecting Changes

The detection part of our method, shown in Algorithm 1, continuously processes the cases it receives. For each observation $\boldsymbol{d}$ and node $X$, the method measures (using CONFLICTMEASURE($\mathcal{B}$, $X$, $\boldsymbol{d}$)) how well $\boldsymbol{d}$ fits with the local structure of $\mathcal{B}$ around $X$. Based on the history of measurements for node $X$, $\boldsymbol{c}_X$, the method tests (using SHIFTINSTREAM($\boldsymbol{c}_X$, $k$)) whether a shift occurred $k$ observations ago. $k$ thus acts as the number of observations that are allowed to "pass" before the method should realize that a shift has taken place. We therefore call the parameter $k$ the *allowed delay* of the method. When the actual detection has taken place, as a last step, the detection algorithm invokes the updating algorithm (UPDATENET($\cdot$)) with the set of nodes, for which SHIFTINSTREAM($\cdot$) detected a change, together with the last $k$ observations.

**Algorithm 1** *Algorithm for BN adaption. Takes as input an initial network $\mathcal{B}$, defined over variables $\boldsymbol{V}$, a series of cases $\mathcal{D}$, and an allowed delay $k$ for detecting shifts in $\mathcal{D}$.*

```
 1: procedure ADAPT(B, V, D, k)
 2:     D' ← []
 3:     c_X ← []   (∀X ∈ V)
 4:     loop
 5:         d ←NEXTCASE(D)
 6:         APPEND(D', (d))
 7:         S ← ∅
 8:         for X ∈ V do
 9:             c ←CONFLICTMEASURE(B, X, d)
10:             APPEND(c_X, c)
11:             if SHIFTINSTREAM(c_X, k) then
12:                 S ← S ∪ {X}
13:         D' ←LASTKENTRIES(D', k)
14:         if S ≠ ∅ then
15:             UPDATENET(B, S, D')
```

To monitor how well each observation $\boldsymbol{d} \equiv (d_1, \ldots, d_m)$ "fit" the current model $\mathcal{B}$, and especially the connections between a node $X_i$ and the remaining nodes in $\mathcal{B}$, we have followed the approach of Jensen et al. (1991): If the current model is correct, then we would in general expect that the probability for observing $\boldsymbol{d}$ dictated by $\mathcal{B}$ is higher than or equal to that yielded by most other models. This should especially be the case for the empty model $\mathcal{E}$, where all nodes are unconnected. That is, in $\mathcal{B}$, we expect the individual attributes of $\boldsymbol{d}$ to be positively correlated (unless $\boldsymbol{d}$ is a rare case, in which case all bets are off):

$$\log \frac{P_{\mathcal{E}}(X_i = d_i)}{P_{\mathcal{B}}(X_i = d_i | X_j = d_j \ (\forall j \neq i))} > 0 \ . \quad (1)$$

Therefore, we let CONFLICTMEASURE($\mathcal{B}$, $X_i$, $\boldsymbol{d}$) return the value given on the left-hand side of (1). We note that this is where the assumption of complete data comes into play: If $\boldsymbol{d}$ is not completely observed, then (1) cannot be evaluated for all nodes $X_i$.

Since a high value returned by CONFLICTMEASURE($\cdot$) for a node $X$ could be caused by a rare case, we cannot use that value directly for determining whether a shift has occurred. Rather, we look at the block of values for the last $k$ cases, and compare these with those from before that. If there is a tendency towards higher values in the former, then we conclude that this cannot be caused only by rare cases, and that a shift must have occurred. Specifically, for each variable $X$, SHIFTINSTREAM($\boldsymbol{c}_X$, $k$) checks whether there is a significant increase in the values of the last $k$ entries in $\boldsymbol{c}_X$ relative to those before that. In our implementation SHIFTINSTREAM($\boldsymbol{c}_X$, $k$) calculates the negative of the second discrete cosine transform component (see e.g. (Press et al., 2002)) of the last $2k$ measures in $\boldsymbol{c}_X$, and returns true if this statistic exceeds a pre-specified threshold value. We are unaware of any previous work using this technique for change point detection, but we chose to use this as it outperformed the more traditional methods of log-odds ratios and $t$-tests in our setting.

### 4.2 Learning and Incorporating Changes

When a shift involving nodes $\boldsymbol{S}$ has been detected, UPDATENET($\mathcal{B}$, $\boldsymbol{S}$, $\mathcal{D}'$) in Algorithm 2 adapts the BN $\mathcal{B}$ around the nodes in $\boldsymbol{S}$ to fit the empirical distribution defined by the last $k$ cases $\mathcal{D}'$ read from $\mathcal{D}$. Throughout the text, both the cases and the empirical distribution will be denoted $\mathcal{D}'$. Since we want to reuse the knowledge encoded in $\mathcal{B}$ that has not been deemed outdated by the detection part of the method, we will update $\mathcal{B}$ to fit $\mathcal{D}'$ based on the assumption that only nodes in $\boldsymbol{S}$ need updating of their probabilistic bindings (i.e. the structure associated with their Markov boundaries in $\mathcal{B}_{\mathcal{D}'}$). Ignoring most details for the moment, the updating method in Algorithm 2 first runs through the nodes

**Algorithm 2** *Update Algorithm for BN. Takes as input the network to be updated $\mathcal{B}$, a set of variables whose structural bindings may be wrong $\boldsymbol{S}$, and data to learn from $\mathcal{D}'$.*

```
1: procedure UpdateNet(B, S, D')
2:     for X ∈ S do
3:         MB̂_D'(X) ← MarkovBoundary(X, D')
4:         G_X ← Adjacencies(X, MB̂_D'(X), D')
5:         PartiallyDirect(X, G_X, MB̂_D'(X), D')
6:     G' ← MergeFragments({G_X}_{X∈S})
7:     G'' ← MergeConnections(B, G', S)
8:     (G'', C) ← Direct(G'', B, S)
9:     Φ'' ← ∅
10:    for X ∈ V do
11:        if X ∈ C then
12:            Φ'' ← Φ'' ∪ {P_D'(X|PA_G''(X))}
13:        else
14:            Φ'' ← Φ'' ∪ {P_B(X|PA_G''(X))}
15:    B ← (G'', Φ'')
```

in $\boldsymbol{S}$ and learns a partially directed graph fragment $\mathcal{G}_X$ for each node $X$ ($\mathcal{G}_X$ can roughly be thought of as a "local completed pattern" for $X$). When network fragments have been constructed for all nodes in $\boldsymbol{S}$, these fragments are merged into a single graph $\mathcal{G}'$, which is again merged with fragments from the original graph of $\mathcal{B}$. The merged graph is then directed using four direction rules, which try to preserve as much of $\mathcal{B}$'s structure as possible, without violating the newly uncovered knowledge represented by the learned graph fragments. Finally, new CPTs are constructed for those nodes $\boldsymbol{C}$ that have a new parent set in $\mathcal{B}_{\mathcal{D}'}$ (nodes which, ideally, should be a subset of $\boldsymbol{S}$).

The actual construction of $\mathcal{G}_X$ is divided into three steps: First, an estimate $\widehat{\mathbf{MB}}_{\mathcal{D}'}(X)$ of $\mathbf{MB}_{\mathcal{D}'}(X)$ is computed, using MarkovBoundary($X$, $\mathcal{D}'$); second, nodes in $\widehat{\mathbf{MB}}_{\mathcal{D}'}(X)$ that are adjacent to $X$ in $\mathcal{B}_{\mathcal{D}'}$ are uncovered, using Adjacencies($X$, $\widehat{\mathbf{MB}}_{\mathcal{D}'}(X)$, $\mathcal{D}'$), and $\mathcal{G}_X$ is initialized as a graph over $X$ and these nodes, where $X$ is connected with links to these adjacent nodes; and third, some links in $\mathcal{G}_X$ that are arcs in $\mathcal{B}^{**}_{\mathcal{D}'}$ are directed as they would be in $\mathcal{B}^{**}_{\mathcal{D}'}$ using PartiallyDirect($X$, $\mathcal{G}_X$, $\widehat{\mathbf{MB}}_{\mathcal{D}'}(X)$, $\mathcal{D}'$). See (Nielsen and Nielsen, 2006) for more elaboration on this.

In our experimental implementation, we used the decision tree learning method of Frey et al. (2003) to find $\widehat{\mathbf{MB}}_{\mathcal{D}'}(X)$, and the AlgorithmPCD($\cdot$) method in (Peña et al., 2005) (restricted to the variables in $\widehat{\mathbf{MB}}_{\mathcal{D}'}(X)$) to find variables adjacent to $X$. The latter method uses a greedy search for iteratively growing and shrinking the estimated set of adjacent variables until no further change takes place. Both of these methods need an "independence oracle" $I_{\mathcal{D}'}$. For this we have used a $\chi^2$ test on $\mathcal{D}'$.

**Algorithm 3** *Uncovers the direction of some arcs adjacent to a variable $X$ as they would be in $\mathcal{B}^{**}_{\mathcal{D}'}$. $\mathbf{NE}_{\mathcal{G}_X}(X)$ consists of the nodes connected to $X$ by a link in $\mathcal{G}_X$.*

```
1: procedure PartiallyDirect(X, G_X, MB̂_D'(X), D')
2:     DirectAsInOtherFragments(G_X)
3:     for Y ∈ MB̂_D'(X) \ (NE_G_X(X) ∪ PA_G_X(X)) do
4:         for T ⊊ MB̂_D'(X) \ {X, Y} do
5:             if I_D'(X, Y | T) then
6:                 for Z ∈ NE_G_X(X) \ T do
7:                     if ¬I_D'(X, Y | T ∪ {Z}) then
8:                         LinkToArc(G_X, X, Z)
```

The method PartiallyDirect($X$, $\mathcal{G}_X$, $\widehat{\mathbf{MB}}_{\mathcal{D}'}(X)$, $\mathcal{D}'$) directs a number of links in the graph fragment $\mathcal{G}_X$ in accordance with the direction of these in (the unknown) $\mathcal{B}^{**}_{\mathcal{D}'}$. With DirectAsInOtherFragments($\mathcal{G}_X$), the procedure first exchanges links for arcs, when previously directed graph fragments unanimously dictate this. The procedure then runs through each variable $Y$ in $\widehat{\mathbf{MB}}_{\mathcal{D}'}(X)$ not adjacent to $X$, finds a set of nodes in $\widehat{\mathbf{MB}}_{\mathcal{D}'}(X)$ that separate $X$ from $Y$, and then tries to re-establish connection to $Y$ by repeatedly expanding the set of separating nodes by a single node adjacent to $X$. If such a node can be found it has to be a child of $X$ in the completed pattern $\mathcal{B}^{**}_{\mathcal{D}'}$, and no arc in the pattern $\mathcal{B}^{*}_{\mathcal{D}'}$ originating from $X$ is left as a link by the procedure (see (Nielsen and Nielsen, 2006) for proofs). As before the independence oracle $I_{\mathcal{D}'}$ was implemented as a $\chi^2$ test in our experiments.

In most constraint based learning methods, only the direction of arcs participating in v-structures are uncovered using independence tests, and structural rules are relied on for directing the remaining arcs afterwards. For the proposed method, however, more arcs from the completed pattern, than just those of v-structures, are directed through independence tests. The reason is that traditional uncovering of the direction of arcs in a v-structure $X \rightarrow Y \leftarrow Z$ relies not only on knowledge that $X$

and $Y$ are adjacent, and that $X$ and $Z$ are not, but also on the knowledge that $Y$ and $Z$ are adjacent. At the point, where $\mathcal{G}_X$ is learned, however, knowledge of the connections among nodes adjacent to $X$ is not known (and may be dictated by $\mathcal{D}'$ or may be dictated by $\mathcal{B}$), so this traditional approach is not possible. Of course these unknown connections could be uncovered from $\mathcal{D}'$ using a constraint based algorithm, but the entire point of the method is to avoid learning of the complete new network.

When all graph fragments for nodes in $\boldsymbol{S}$ have been constructed, they are merged through a simple graph union in MERGEFRAGMENTS($\cdot$); no conflicts among orientations can happen due to the construction of PARTIALLYDIRECT($\cdot$). In MERGECONNECTIONS($\mathcal{B}$, $\mathcal{G}'$, $\boldsymbol{S}$) connections among nodes in $\boldsymbol{V} \setminus \boldsymbol{S}$ are added according to the following rule: If $X, Y \in \boldsymbol{V} \setminus \boldsymbol{S}$ are adjacent in $\mathcal{B}$, then add the link $X - Y$ to $\mathcal{G}'$. The reason for this rule is that inseparable nodes, for which no change has been detected, are assumed to be inseparable still. However, the direction of some of the arcs may have changed in $\mathcal{G}'$, wherefore we cannot directly transfer the directions in $\mathcal{B}$ to $\mathcal{G}'$.

Following the merge, DIRECT($\mathcal{G}''$, $\mathcal{B}$, $\boldsymbol{S}$) directs the remaining links in $\mathcal{G}''$ according to the following five rules:

1. If $X \in \boldsymbol{V} \setminus \boldsymbol{S}$, $X - Y$ is a link, $X \to Y$ is an arc in $\mathcal{B}$, and $Y$ is a descendant of some node $Z$ in $\mathbf{MB}_{\mathcal{B}}(X) \setminus \mathbf{AD}_{\mathcal{B}}(X)$, where $\mathbf{AD}_{\mathcal{B}}(X)$ are nodes adjacent to $X$ in $\mathcal{B}$, through a path involving only children of $X$, then direct the link $X - Y$ as $X \to Y$.[3]

2. If Rule 1 cannot be applied, and if $X - Y$ is a link, $Z \to X$ is an arc, and $Z$ and $Y$ are non-adjacent, then direct the link $X - Y$ as $X \to Y$.

3. If Rule 1 cannot be applied, and if $X - Y$ is a link and there is a directed path from $X$ to $Y$, then direct the link $X - Y$ as $X \to Y$.

4. If Rules 1 to 3 cannot be applied, chose a link $X - Y$ at random, such that $X, Y \in \boldsymbol{V} \setminus \boldsymbol{S}$, and direct it as in $\mathcal{B}$.

5. If Rules 1 to 4 cannot be applied, chose a link at random, and direct it randomly.

Due to potentially flawed statistical tests, the resultant graph may contain cycles each involving at least one node in $\boldsymbol{S}$. These are eliminated by reversing only arcs connecting to at least one node in $\boldsymbol{S}$. The reversal process resembles the one used in (Margaritis and Thrun, 2000): We remove all arcs connecting to nodes in $\boldsymbol{S}$ that appears in at least one cycle. We order the removed arcs according to how many cycles they appear in, and then insert them back in the graph, starting with the arcs that appear in the least number of cycles, breaking ties arbitrarily. When at some point the insertion of an arc gives rise to a cycle, we insert the arc as its reverse.

We have obtained a proof of the "correctness" of the proposed method, but space restrictions prevents us from bringing it here. Basically, we have shown that, given the set-up from Section 3, if the method is started with a network equivalent to $\mathcal{B}_{P_1}$, then $\mathcal{B}_i$ will be equivalent to $\mathcal{B}_{P_{i-k}}$ for all $i > k$. This is what we refer to as "correct" behaviour, and it means that once on the right track, the method will continue to adapt to the underlying distribution, with the delay $k$. The assumptions behind the result, besides that each $P_i$ is DAG faithful, are i) the samples in $\mathcal{D}$ are representative of the distributions they are drawn from, ii) the rank of $\mathcal{D}$ is bigger than $2k$, and iii) SHIFTINSTREAM($\cdot$) returns true for variable $X$ and sample $j$ iff $P_{j-k}$ is not similar to $P_{j-k-1}$ around $X$ (see (Nielsen and Nielsen, 2006) for formal definitions and detailed proofs).

## 5 Experiments and Results

To investigate how our method behaves in practice, we ran a series of experiments. We constructed 100 experiments, where each consisted of five randomly generated BNs $\mathcal{B}_1, \ldots, \mathcal{B}_5$ over ten variables, each having between two and five states. We made sure that $\mathcal{B}_i$ was structurally identical to $\mathcal{B}_{i-1}$ except for the connection between two randomly chosen nodes. All CPTs in $\mathcal{B}_i$ were kept the same as in $\mathcal{B}_{i-1}$, except for the nodes with a new parent

[3] That Rule 1 is sensible is proved in (Nielsen and Nielsen, 2006). Intuitively, we try to identify a graph fragment for $X$ in $\mathcal{B}$, that can be merged with the graph fragments learned from $\mathcal{D}'$. It turns out that the arcs directed by Rule 1 are exactly those that would have been learned by PARTIALLYDIRECT($X$, $\mathcal{G}_X$, $\mathbf{MB}_{\mathcal{B}}(X)$, $P_{\mathcal{B}}$).

set. For these we employed four different methods for generating new distributions: A estimated the probabilities from the previous network with some added noise to ensure that no two distributions were the same. B, C, and D generated entirely new CPTs, with B drawing distributions from a uniform distribution over distributions. C drew distributions from the same distribution, but rejected those CPTs where there were no two parent configurations, for which the listed distributions had a KL-distance of more than $1$. D was identical to C, except for having a threshold of $5$. The purpose of the latter two methods is to ensure strong probabilistic dependencies for at least one parent configuration. For generation of the initial BNs we used the method of Ide et al. (). For each series of five BNs, we sampled $r$ cases from each network and concatenated them into a piecewise DAG faithful sample sequence of rank $r$ and length $5r$, for $r$ being $500$, $1000$, $5000$, and $10000$.

We fed our method (NN) with the generated sequences, using different delays $k$ ($100$, $500$, and $1000$), and measured the deciance wrt. the KL-distance on each. As mentioned we are unaware of other work geared towards non-stationary distributions, but for base-line comparison purposes, we implemented the structural adaptation methods of Friedman and Goldszmidt (1997) (FG) and Lam and Bacchus (1994) (LB). For the method of Friedman and Goldszmidt (1997) we tried both simulated annealing (FG-SA) and a more time consuming hill-climbing (FG-HC) for the unspecified search step of the algorithm. As these methods have not been developed to deal with non-stationary distributions, they have to be told the delay between learning. For this we used the same value $k$, that we use as delay for our own method, as this ensure that all methods store only a maximum of $k$ full cases at any one time. The chosen $k$ values, also correspond to those found for the experimental results reported in Friedman and Goldszmidt (1997) and Lam (1998). The only other pre-specified parameter required by our method, viz. a threshold for the $\chi^2$-tests we set at a conventional $0.05$. Each method was given the correct initial network $\mathcal{B}_1$ to start its exploration.

Space does not permit us to present the results in full, but the deviance of both NN, FG-SA, and FG-HC are presented in Figures 1 and 2. NN outperformed FG-SA in 81 of the experiments, and FG-HC in 65 of the experiments. The deviance of the LB method was much worse than for either of these three. The one experiment, where both the FG methods outperformed the NN method substantially, had $r$ equal to $10000$ and $k$ equal to $1000$, and was thus the experiment closest to the assumption on stationary distributions of the FG and LB learners.

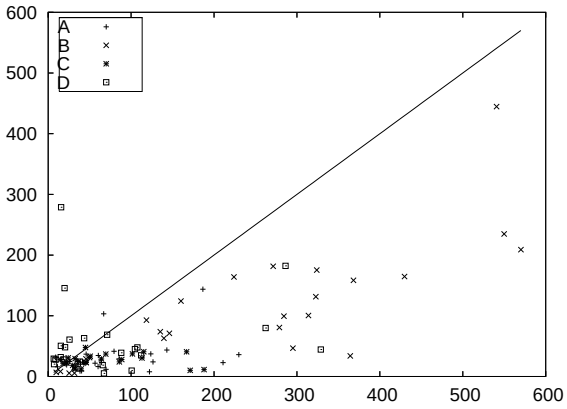



Figure 1: Deviance measures FG-SA (X-axis) vs. NN (Y-axis).

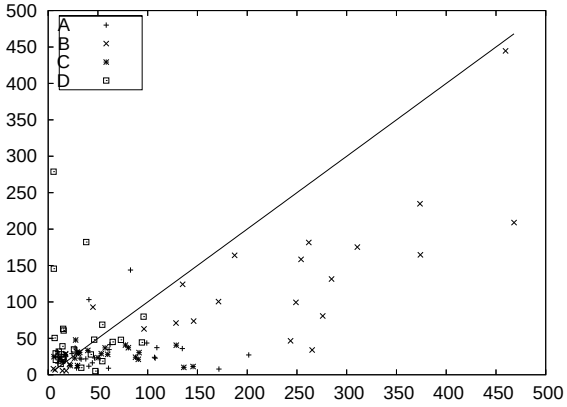



Figure 2: Deviance measures FG-HC (X-axis) vs. NN (Y-axis).

Studying the individual experiments more closely, it became apparent that NN is more "stable" than FG: It does not alter the network as often as FG, and when doing so, NN does not alter it as much as FG. This is positive, as besides preventing unnecessary computations, it frees the user of

the learned nets from a range of false positives. Furthermore, we observed that the distance between the BN maintained by NN and the generating one seems to stabilize after some time. This was not always the case for FG.

## 6 Discussion

The plotted experiments seem to indicate that our method is superior to existing techniques for domains, where the underlying distribution is not stationary. This is especially underscored by our experiments actually favouring the existing methods through using only sequences whose rank is a multiplum of the learners $k$ value, which means that both FG and LB always learn from data from only one partition of the sequence, unlike NN, which rarely identifies the point of change completely accurately. Moreover, score based approaches are geared towards getting small KL-scores, and thus the metric we have reported should favour FG and LB too.

Of immediate interest to us, is investigation of how our method fares when the BN given to it at the beginning is not representative of the distribution generating the first partition of the sample sequence. Also, it would be interesting to investigate the extreme cases of sequences of size 1 and those with very low ranks ($r \lll 500$). Obviously, the task of testing other parameter choices and other implementation options for the helper functions need to be carried out too.

Currently, we have some ideas for optimizing the precision of our method, including performing parameter adaptation of the CPTs associated with the maintained structure, and letting changes "cascade", by marking nodes adjacent to changed nodes as changed themselves.

In the future it would be interesting to see how a score based approach to the local learning part of our method would perform. The problem with taking this road is that it does not seem to have any formal underpinnings, as the measures score based approaches optimize are all defined in terms of a single underlying distribution. A difficulty which Friedman and Goldszmidt (1997) also allude to in their efforts to justify learning from data collections of varying size for local parts of the network.

## References


W. Buntine. 1991. Theory refinement on Bayesian networks. In *UAI 91*, pages 52–60. Morgan Kaufmann.

L. Frey, D. Fisher, I. Tsamardinos, C. F. Aliferis, and A. Statnikov. 2003. Identifying Markov blankets with decision tree induction. In *ICDM 03*, pages 59–66. IEEE Computer Society Press.

N. Friedman and M. Goldszmidt. 1997. Sequential update of Bayesian network structure. In *UAI 97*, pages 165–174. Morgan Kaufmann.

D. Heckerman. 1998. A tutorial on learning with Bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*, pages 301–354. Kluwer.

J. S. Ide, F. G. Cozman, and F. T. Ramos. Generating random Bayesian networks with constraints on induced width. In *ECAI 04*, pages 323–327. IOS Press.

F. V. Jensen, B. Chamberlain, T. Nordahl, and F. Jensen. 1991. Analysis in HUGIN of data conflict. In *UAI 91*. Elsevier.

W. Lam and F. Bacchus. 1994. Using new data to refine a Bayesian network. In *UAI 94*, pages 383–390. Morgan Kaufmann.

W. Lam. 1998. Bayesian network refinement via machine learning approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):240–251.

D. Margaritis and S. Thrun. 2000. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 12*, pages 505–511. MIT Press.

S. H. Nielsen and T. D. Nielsen. 2006. Adapting bayes nets to non-stationary probability distributions. Technical report, Aalborg University. www.cs.aau.dk/~holbech/nielsennielsen_note.ps.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

J. M. Peña, J. Björkegren, and J. Tegnér. 2005. Scalable, efficient and correct learning of Markov boundaries under the faithfulness assumption. In *ECSQARU 05*, volume 3571 of *LNCS*, pages 136–147. Springer.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, editors. 2002. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2nd edition.

J. Roure. 2004. Incremental hill-climbing search applied to Bayesian network structure learning. In *Proceedings of the 15th European Conference on Machine Learning*. Springer.

T. Verma and J. Pearl. 1990. Equivalence and synthesis of causal models. In *UAI 91*, pages 220–227. Elsevier.

# Diagnosing Lyme disease - Tailoring patient specific Bayesian networks for temporal reasoning

Kristian G. Olesen (kgo@cs.aau.dk)
Department of Computer Science, Aalborg University
Fredrik Bajersvej 7 E, DK-9220 Aalborg East, Denmark

Ole K. Hejlesen (okh@mi.aau.dk)
Department of Medical Informatics, Aalborg University
Fredrik Bajersvej 7 D, DK-9220 Aalborg East, Denmark

Ram Dessau (rde@cn.stam.dk)
Clinical Microbiology, Næstved Hospital
Denmark

Ivan Beltoft (ibe@netcompany.com) and Michael Trangeled (michael@trangeled.com)
Department of Medical Informatics, Aalborg University

**Abstract**

Lyme disease is an infection evolving in three stages. Lyme disease is characterised by a number of symptoms whose manifestations evolve over time. In order to correctly classify the disease it is important to include the clinical history of the patient. Consultations are typically scattered at non-equidistant points in time and the probability of observing symptoms depend on the time since the disease was inflicted on the patient.

A simple model of the evolution of symptoms over time forms the basis of a dynamically tailored model that describes a specific patient. The time of infliction of the disease is estimated by a model search that identifies the most probable model for the patient given the pattern of symptom manifestations over time.

## 1 Introduction

Lyme disease, or Lyme Borreliosis, is the most commonly reported tick-borne infection in Europe and North America. Lyme disease was named in 1977 when a cluster of children in and around Lyme, Connecticut, displayed a similar disease pattern. Subsequent studies revealed that the children were infected by a bacteria, which was named Borrelia burgdorferi. These bacteria are transmitted to humans by the bite of infected ticks (Ixodus species). The clinical manifestations of Lyme borreliosis are described in three stages and may affect different organ systems, most frequently the skin and nervous system. The diagnosis is based on the clinical findings and results of laboratory testing for antibodies (immunglobulin M and G) in the blood. Methods for direct detection of the bacteria in the tissue or the blood are not available for routine diagnosis. Thus it may be difficult to diagnose Lyme disease as the symptoms may not be exclusive for the disease and testing for antibodies may yield false positive or negative results.

In the next section we give an overview of Lyme disease and in section 3 a brief summary of two earlier systems are given. In section 4 we suggest an approach that overcome the difficulties by modelling multiple consultations spread out in non-equidistant points in time. This approach assumes that the time of infection is known, but this is rarely the case. We aim for an estimation of this fact; a number of hypothesised models are generated and the best one is

| **Organ system** | **Stage 1:** Incubation period one week (few days to one month) | **Stage 2:** Incubation period one week to a few months | **Stage 3:** Incubation period months to years |
|---|---|---|---|
| Skin | Erythema migrans | Borrelial lympocytoma | Acrodermatitis chronica athrophicans |
| Central nervous system | | Early neuroborreliosis | Chronic neuroborreliosis |
| Joints | | | Lyme arthritis |
| Heart | | Lyme carditis | |
| Average sentitivity of antibody detection (IgG or IgM) | 50% | 80% | 100% |

Table 1: Clinical manifestations of Lyme disease.

identified based on the available evidence. In section 5 a preliminary evaluation of the approach is described and finally we conclude in section 6.

## 2 Lyme Disease

According to European case defintions (http://www.oeghmp.at/eucalb/diagnosis_case-definition-outline.html) Lyme disease and its manifestations are described in three stages as shown in Table 1.

Erythema migrans is the most frequent manifestation. The rash which migrates from the site of the tickbite continuously grows in size. Erythema migrans is fairly characteristic and the diagnosis is clinical. Laboratory testing is useless as only half of the patients are positive, when the disease is localized to the skin only. Neuroborreliosis is less frequent, in Denmark the incidence is around 3/100.000 per year. The IgM or IgG may be positive in 80% of patients with neuroborreliosis, but if the duration of the the clinical disease is more than two months, then 100% of patients are positive. Figure 1 shows that not only the probability of being positive, but also the level of antibodies vary as a function of time. Thus the laboratory measurement of IgM and IgG antibodies depends both on the duration of the clinical disease and the dissemination of the infection to other organ systems than the skin. The incidence of the other manifestations of Lyme disease is more rare. The diagnosis of lyme arthritis is especially difficult as the symptoms are similar to arthritis due to other causes and because the disease is rare.

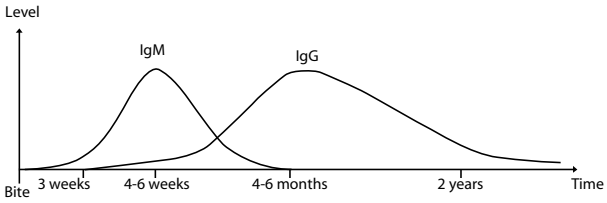



Figure 1: IgM and IgG development after infection of Lyme disease.

The epidemiology is complex. For example different age groups are at different risks, there is a large seasonsal variation due to variations in tick activity (Figure 2), the incubation period may vary from a few days to several years and some clinical manifestations are very rare. There are large variations in incubation time and clinical progression. Some patients may have the disease starting with a rash (erythema migrans) and then progressing to ntaeuroborreliosis, but most patients with neuroborreliosis are not aware of a preceeding rash. The description of the disease and it's progression primarily based on (Smith et al., 2003) and (Gray et al., 2002).

There are also large individual variations in

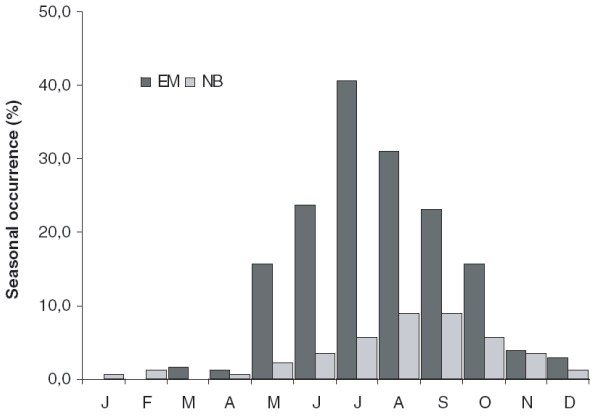



Figure 2: Comparison of seasonal occurrence of erythema migrans (EM, n = 566) and neuroborreliosis (NB, n = 148) in northeastern Austria, 1994-1995.(Gray et al., 2002).

the behaviour of the patient and the physician, about when to visit the doctor, when to take the laboratory test ect. Possible repeated visits to the doctor and repeated laboratory testing are performed at variable intervals. Joint pain and arthritis is a very common problem in the general population, but is only very rarely due to Lyme borreliosis. However, many patients with arthritis are tested for Lyme borreliosis ( 30% of all patients samples) and as the prior probability of Lyme disease is low, the posterior probality of Lyme arthritis is also low in spite of a positive test result. Laboratories in some countries attempt to improve the diagnosis of Lyme disease by using a two-step strategy in antibody detection, by setting a low cut-off on the screening test and then performing a more expensive and complex test to confirm or discard the result of the primary test. As the second test is using the same principle of indirect testing for antibody reactions to different Borrelial antigens the two tests are highly correlated, and the information gain is limited. Thus, the basic problem of false positive or false negative results is not solved. It was therefore found important to develop a decision support system to assist the clinician by calculating the posterior probability of Lyme disease to guide the choice of treatment. An evidence based clinical diagnosis is supported by incorporating clinical and laboratory data into the model.

To capture the complex patterns of the disease, a model must incorporate the relevant clinical evidence including estimation of the temporal aspects of time since the tickbite, the duration of clinical disease and the development of antibody response.

## 3 Existing Models for Diagnosis of Lyme Disease

We have knowledge of two models that have been developed to assist the medical practitioner in the diagnosis of Lyme disease (Dessau and Andersen, 2001; Fisker et al., 2002). Both models are based on Bayesian networks (Jensen, 2001) and as the latter is a further development of the former, they share most of the variables.

The models include a group of variables describing general information and knowledge including age and gender of the patient, the patient's exposure to ticks (is the patient a forrest worker or orienteer), the month of the year and whether the patient recalls a tick bite. The information is used to establish whether or not the conditions for a Lyme disease infection has been present. This part of the model influence the hypothesis variable, Borrelia.

Another section of the models describe clinical manifestations. The findings are influenced by the hypothesis variable, but may be caused by other reasons. Similarly, a section of variables describing laboratory findings may be caused by either Lyme disease or by other disorders.

The structure of the model by Dessau and Andersen (2001) is shown in Figure 3. In this model the three stages of Lyme disease is explicitly represented as three copies of the hypothesis and the findings sections. The conditional probabilities reflect the temporal evolution of symptoms, such that e.g. neuroborreliosis is more probable in stage two than in stages one and three. A problem with this approach is that the progression of the disease is uncertain, and consequently it is unclear which copy of e.g. finding

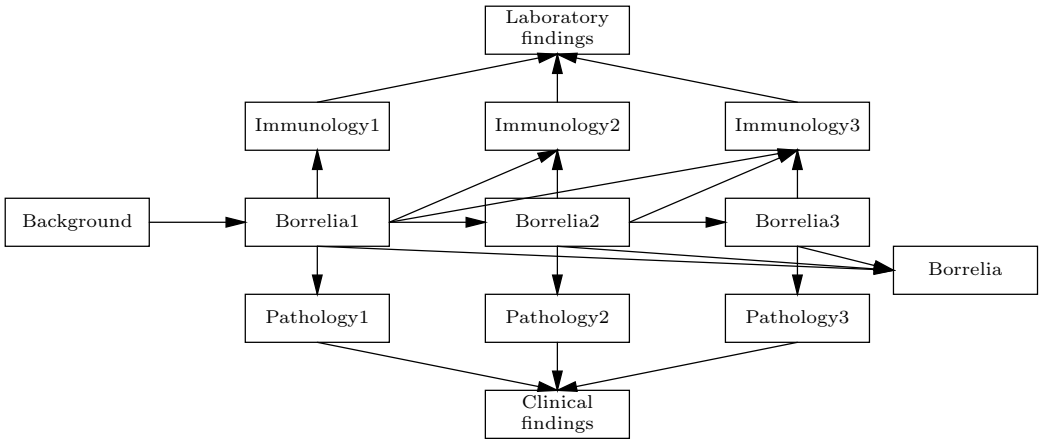



Figure 3: Structure of the Bayesian network by Dessau and Andersen.

variables to use. This was resolved by combining variables from each stage in single variables that describe the observable clinical manifestations and laboratory findings.

The progress of Lyme disease was modelled by a variable "duration" (not shown in the figure), indicating the time a patient has experienced the symptoms. It does not distinguish between the duration of the different symptoms, but models the total duration of illness. If the duration is short it indicates a stage one Lyme disease and so forth.

Dessau and Andersen's model is a snapshot of a patient at a specific point in time and it does not involve temporal aspects such as means for entering of multiple evidence on the same variables as a result of repeated consultations.

The model by Fisker et al. (2002) aims to include these issues. The idea is to reflect the clinical practice to examine a patient more than once if the diagnosis is uncertain. This gives the medical practitioner the opportunity to observe if the development of the disease corresponds to the typical pattern. The model mainly includes the same variables, but the structure of the network is different (see Figure 4).

Instead of triplicating the pathology for the different stages of the disease the temporal aspect was incorporated in the model by defining the states of manifestation variables as time intervals. This approach was inspired by (Arroyo-Figueroa and Sucar, 1999) that introduced Temporal Nodes Bayesian Networks as an alternative to dynamic Bayesian networks. The time intervals represent the time since the symptom was first observed, and both the length and the number of time intervals was tailored individually for each node. With this approach it became possible to enter "old" evidence into the model. For example, if erythema migrans was observed on, or recalled by, the patient ten weeks ago and lymphocytoma is a current observation, the *EM* variable is instantiated to the state *8 weeks - 4 months* and the *lymphocy* is instantiated to $<$ *6 weeks*.

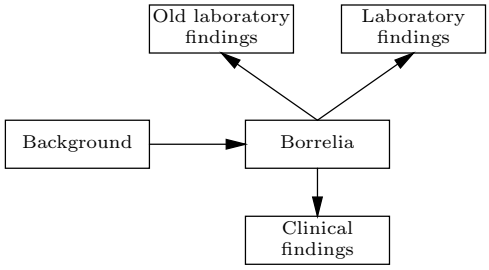



Figure 4: Structure of the Bayesian network by Fisker et al.

The model also incorporated the ability to enter duplicate results of laboratory tests. This option was included by repeating the laboratory findings and including a variable specifying the time between the result of the old test and the current test (not shown in the figure).

The two models basically use the same features but differ in the modelling of the progres-

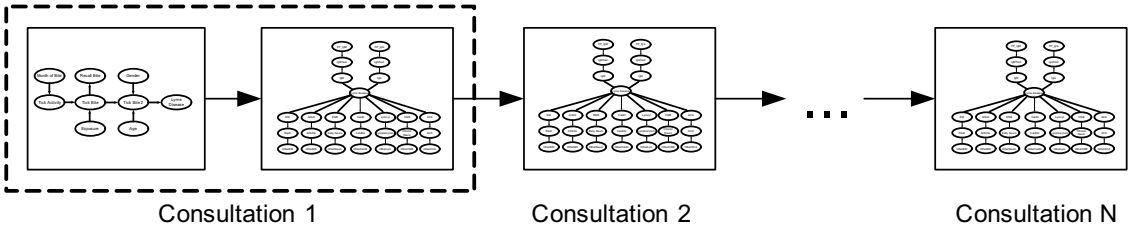



Figure 5: The structure of the tailored patient specific model is composed of background knowledge and consultation modules.

sion of the disease. Further, the model by Fisker et al. takes the use of data from previous consultations into consideration.

Both models implement the possibility of reading the probability of having Lyme disease in one of the three stages, but with two different approaches. The model by Fisker et al. models the stages as states in a single variable, whereas the model by Dessau and Andersen models the stages explicitly as causally dependent variables.

The main problem in both models is the temporal progression of Lyme disease. Dessau and Andersen's model gives a static picture of the current situation, that only takes the total duration of the period of illness into consideration. Fisker et al. is more explicit in the modelling of the progress of the disease by letting states denote time intervals since a symptom was first observed and by duplicating variables for laboratory findings. This enables inclusion of findings from the past, but is still limited to single observations for clinical evidence and two entries for laboratory tests.

Besides the difficulties in determining the structure of the models, a considerable effort was required to specify the conditional probabilities.

In the following section we propose a further elaboration of the modelling by introduction of continuous time and an arbitrary number of consultations.

## 4 Tailoring patient specific models

We aim for a Bayesian network that incorporates the full clinical history of individual patients. The model is composed of modules, where each module describes a consultation. The problem is that consultations may appear at random points in time and that the conditional probabilities for the symptoms at each consultation vary, depending on the time that has passed since the tick bite. Therefore frameworks such as dynamic Bayesian networks and Markov chains do not apply. We introduce continuous time inspired by (Nodelman et al., 2002). The proposed approach involve a model for the conditional probabilities, but before reaching that point we determine the structure of the model.

### 4.1 Structure

The structure of the patient specific model is illustrated in Figure 5. At the first consultation the background knowledge module is linked to a generic consultation module consisting of the disease node, clinical findings and laboratory findings. In order to keep focus on the overall modelling technique we deliberately kept the model simple. The consultation module is a naive Bayes model, and subsequent consultations are included in the the model by extendeding it with a new consultation module. This process can be repeated for an arbitrary number of consultations. The consultation modules are connected only through the disease nodes. This is a quite strong assumption that may be debated, but it simplifies things and keep the complexity of the resulting model linear in the number of consultations. Less critical is that the disease nodes do not take the stage of the disease into account; we consider that the classification into stages are mostly for descriptive purposes, but it could be modeled as stages in

the disease node, although this would complicate the quantitative specification slightly.

### 4.2 Conditional probability tables

As the symptoms of Lyme disease vary over time we assume a simple model for the conditional probabilities in the model. We approximate the conditional probabilities for the symptoms by a continuous function, composed as a mixture of two sigmoid functions. This is a somewhat arbitrary choise; other models may be investigated, but for the present purpose this simple model suffice. The functions describing the conditional probabilities are based on empirical knowledge. From existing databases we extract the probabilities for the various symptoms at different times since the infection was inflicted on the patient. The time of infliction is usually not known, but around one third of the patients recall a tick bit. In other cases there is indirect evidence for the time of the bite, such as limited periods of exposure.

An example is shown in Figure 6, where the resulting functions for the probability of showing the symptom EM up to 100 days after the infection are drawn. As can be seen, the age of the patient has been incorporporated as a parameter to the function. Thus, we can compute the conditional probabilities for a consultation module directly, provided that the time since the infection is known. This is rarely the case.

### 4.3 Determining the time of infection

The purpose of the Bayesian model is to calculate the probability of Lyme disease based on the clinical findings and possible laboratory evidence. As described in the previous section, the conditional probabilities of the symptoms and the serology are determined by the time since the tick bite. Thus, in order to construct the complete model it is necessary to know the time of infliction.

The clinical history for a given patient is illustrated in Figure 7. Different time intervals are shown in the figure. The interval $t_1$ is the time from the infection to the first consultation, $t_2$ is the interval between the first and the second consultation and so on. When the patient consults the medical practitioner the time of the consultation is registered. Therefore, the intervals between the consultations are known, whereas the interval, $t_1$, from the infection to the first consultation is unknown and must be estimated. It is often difficult to determine $t_1$, because a patient may not recollect a tick bite, and because symptoms may not appear until late in the progress of the disease.

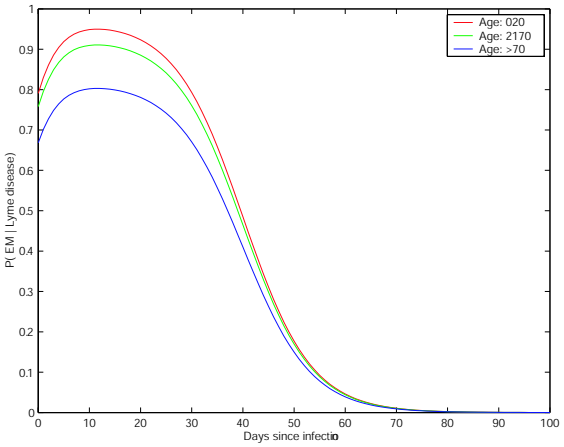



Figure 6: The probability of experiencing erythema migrans (EM) over time modelled as continuous functions. The development of EM is dependent on the age of the patient.

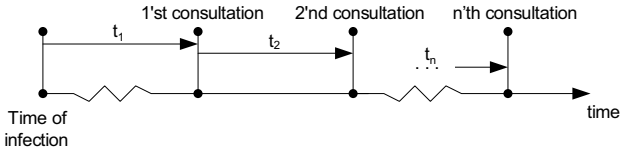



Figure 7: Time line that represents the consultations that forms the clinical history.

As the conditional probabilities for the consultation modules vary depending on $t_1$, different values of $t_1$ will result in Bayesian networks with different quantitative specification. Hence, estimation of $t_1$ is a crucial part of the construction of the patient specific model. We tackle this problem by hypothesising a number of different models, and we identify those that best matches the evidence. Each model includes all consultations for the patient. Thus, we have a fixed structure with different instantiations of

the conditional probabilities for different values of $t_1$.

The probability of a model given the evidence can be calculated by using Bayes' rule:

$$P(M \mid e) = \frac{P(e \mid M) \cdot P(M)}{P(e)} \quad (1)$$

$P(e \mid M)$ can be extracted from the hypothesised Bayesian network as the normalization constant. The probability of the evidence, $P(e)$, is not known, but is a model independent constant. The prior probability of the model, $P(M)$, is assumed to be equal for all models, and is therefore inversely proportional to the number of models. Thus, $P(M \mid e)$ can be expressed by

$$P(M \mid e) \; \alpha \; P(e \mid M) \quad (2)$$

The probability of Lyme disease, $P(Ld)$ can be read from each model and will, of course, vary depending on the model M. By using the fundamental rule, the joint probability of $P(Ld, M \mid e)$ can be obtained as

$$P(Ld, M \mid e) = P(Ld \mid M, e) \cdot P(M \mid e) \quad (3)$$

where $P(M \mid e)$ can be substituted by using equation 2:

$$P(Ld, M \mid e) \; \alpha \; P(Ld \mid M, e) \cdot P(e \mid M) \quad (4)$$

The probability of Lyme disease given the evidence, can now be found by marginalizing over M:

$$P(Ld \mid e) = \sum_{T_n=T_{start}}^{T_{end}} P(Ld \mid M_{T_n}, e) \cdot P(e \mid M_{T_n}) \quad (5)$$

$T_{start}$ and $T_{end}$ represent an interval that surrounds the most probable value of the time since the bite, $t_1$, as illustrated in Figure 8.

The interval is chosen due to computational considerations, and because the assumption that all models are equally probable is obviously not valid. Alternatively, we could simply choose the model with highest probability.

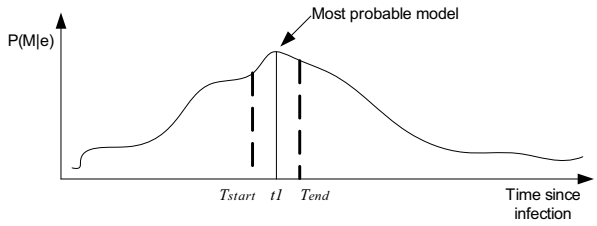



Figure 8: The figure illustrates the probability of the different models as a function of days since infection.

The estimation of the most probable model has been implemented by evaluating all models for $t_1$ in the interval 0-800 days. The upper limit of 800 days is based on the assumption that the general development pattern of Lyme disease at this time is stabilized and the symptoms are chronical. The size of the interval $T_{start}$ - $T_{end}$ has, somewhat arbritrarily, been set to 25 days.

## 5 Preliminary evaluation

In a preliminary test of the proposed approach the results were compared to the outcome of the model by Dessau and Andersen. The evaluation was focused on the structural problems in the models. The cases that were used for the evaluation are based on a survey performed by Ram Dessau where 2414 questionnaires have been collected from the Danish hospitals in Aalborg and Herlev. The overall picture of the comparison is that the proposed model in general estimates a higher probability of Lyme disease than the model by Dessau and Andersen.

In order to evaluate the effect of the incorporation of the clinical history in the time sliced model, a number of typical courses of Lyme disease have been evaluated as single consultations without utilizing the clinical history and as consultations utilizing the previous history. At the same time, each of the consultations have been evaluated in the model by Dessau and Andersen in order to evaluate how it handles the later stage symptoms, when the clinical history is not incorporated.

This informal study indicates that the time sliced model behaves as intended when the clinical history is incorporated. The estimated prob-

ability of Lyme disease is gradually increased for each added consultation, as the general development pattern of the disease is confirmed in typical courses.

In the estimates from the model by Dessau and Andersen the probability of Lyme disease decreases as the later stage symptoms are observed. This reasoning does not seem appropriate when it is known from earlier consultations that the clinical history points toward a borrelial infection. The model by Dessau and Andersen is not designed to incorporate the clinical history, but from the results it can be seen that this parameter is important in order to provide reasonable estimates of the probability of Lyme disease.

## 6 Conclusion

Temporal reasoning is an integral part of medical expertise. Many decisions are based on prognoses or diagnoses where symptoms evolve over time and the effects of treatments typically become apparent only with some delay. A common scenario in the general practice is that a patient attends the clinic at different times on a non-regular basis.

Lyme disease is an infection characterised by a number of symptoms whose manifestations evolve over time. In order to correctly classify the disease it is important to include the clinical history of the patient. We have proposed a method to include consultations scattered over time at non-equidistant points. A description of the evolution of symptoms over time forms the basis of a dynamically tailored model that describes a specific patient. The time of infliction of the disease is estimated by a model search that identifies the most probable model for the patient given the pattern of symptoms over time.

Based on a preliminary evaluation it can be concluded that the method proposed for handling nonequivalent time intervals in order to incorporate the clinical history works satisfactory. In cases where the clinical history confirmed the general development pattern of Lyme disease the estimated probability the disease was gradually increased from consultation to consultation, whereas it was reduced when the history did not confirm the pattern.

We conclude that the proposed method seems viable for temporal domains, where changing conditional probabilities can be modeled by continuous functions.

## References


[Arroyo-Figueroa and Sucar1999] Gustava Arroyo-Figueroa and Luis Enrique Sucar. 1999. A Temporal Bayesian Network for Diagnosis and Prediction. *In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 13–20. ISBN: 1-55860-614-9.

[Dessau and Andersen2001] Ram Dessau and Maj-Britt Nordahl Andersen. 2001. Beslutningsstøtte med Bayesiansk netværk - En model til tolkning af laboratoriesvar. Technical report, Aalborg Universitet. In danish.

[Fisker et al.2002] Brian Fisker, Kristian Kirk, Jimmy Klitgaard, and Lars Reng. 2002. Decision Support System for Lyme Disease. Technical report, Aalborg Universitet.

[Gray et al.2002] J.S. Gray, O. Kahl, R.S. Lane, and G. Stanek. 2002. *Lyme Borreliosis: Biology, Epidemiology and Control*. Cabi Publishing. ISBN: 0-85199-632-9.

[Jensen2001] Finn V. Jensen. 2001. *Bayesian Networks and Decision Graphs*. Springer Verlag. ISBN: 0-387-95259-4.

[Nodelman et al.2002] Uri Nodelman, Christian R. Shelton, and Daphne Koller. 2002. Continuous Time Bayesian Networks. *In Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 378–387. ISBN: 1-55860-897-4.

[Smith et al.2003] Martin Smith, Jeremy Gray, Marta Granström, Crawford Revie, and George Gettinby. 2003. European Concerted Action on Lyme Borreliosis. `http://vie.dis.strath.ac.uk/vie/LymeEU/`. Read 04-12-2003.

# Predictive Maintenance using Dynamic Probabilistic Networks

Demet Özgür-Ünlüakın
Taner Bilgiç
Department of Industrial Engineering
Boğaziçi University
Istanbul, 34342, Turkey

**Abstract**

We study dynamic reliability of systems where system components age with a constant failure rate and there is a budget constraint. We develop a methodology to effectively prepare a predictive maintenance plan of such a system using dynamic Bayesian networks (DBNs). DBN representation allows monitoring the system reliability in a given planning horizon and predicting the system state under different replacement plans. When the system reliability falls below a predetermined threshold value, component replacements are planned such that maintenance budget is not exceeded. The decision of which component(s) to replace is an important issue since it affects future system reliability and consequently the next time to do replacement. Component marginal probabilities given the system state are used to determine which component(s) to replace. Two approaches are proposed in calculating the marginal probabilities of components. The first is a myopic approach where the only evidence belongs to the current planning time. The second is a look-ahead approach where all the subsequent time intervals are included as evidence.

## 1 Introduction

Maintenance can be performed either after the breakdown takes place or before the problem arises. The former is reactive whereas the latter is proactive. Reactive maintenance is appropriate for systems where the failure does not result in serious consequences. Decision-theoretic troubleshooting belongs to this category. Proactive or planned maintenance can be further classified as preventive and predictive (Kothamasu et al., 2006). These differ in the scheduling behaviour. Preventive maintenance performs maintenance on a fixed schedule whereas in predictive maintenance, the schedule is adaptively determined. Reliability-centered maintenance is predictive maintenance where reliability estimates of the system are used to develop a cost-effective schedule.

Decision-theoretic troubleshooting, which balances costs and likelihoods for the best action, is first studied by (Kalagnanam and Henrion, 1988). Heckerman et al.(1995) extend it to the context of Bayesian networks (Pearl, 1988). A similar troubleshooting problem, where multiple but independent faults are allowed, is addressed in (Srinivas, 1995). More recent studies are mostly due to researchers from the SACSO (Systems for Automated Customer Support Operations) project. By assuming a single fault, independent actions and constant costs, and making use of a simple model representation technique (Skaanning et al., 2000), they show that the simple efficiency ordering yields an optimal sequence of actions (Jensen et al., 2001). Langseth and Jensen (2001) present two heuristics for handling dependent actions and conditional costs. Langseth and Jensen (2003) provide a formalism that combines the methodologies used in reliability analysis and decision-theoretic troubleshooting. Koca and Bilgiç (2004) present a generic decision-theoretic troubleshooter to handle troubleshooting tasks incorporating questions, dependent actions, conditional costs, and any combinations of these. Decision-theoretic troubleshooting has always

been studied as a static problem and with an objective to reach a minimum-cost action plan.

On the reliability analysis side, fault diagnosis finds its roots in (Vesely, 1970). Kothamasu et al.(2006) review the philosophies and techniques that focus on improving reliability and reducing unscheduled downtime by monitoring and predicting machine health. Torres-Toledano and Sucar (1998) develop a general methodology for reliability modelling of complex systems based on Bayesian networks. Welch and Thelen (2000) apply DBNs to an example from the dynamic reliability community. Weber and Jouffe (2003, 2006) present a methodology that helps developing DBNs and Dynamic Object Oriented Bayesian Networks (DOOBNs) from available data to formalize reliability of complex dynamic models. Muller et al. (2004) propose a methodology to design a prognosis process taking into account the behavior of environmental events. They develop a probabilistic model based on the translation of a preliminary process model into DBN. Bouillaut et al. (2004) use causal probabilistic networks for the improvement of the maintenance of railway infrastructure. Weber et al. (2004) use DBN to model dependability of systems taking into account degradations and failure modes governed by exogenous constraints.

All of the studies related to reliability analysis using DBNs are *descriptive*. Dynamic problem is represented with DBNs and the outcome of the analysis is how system reliability behaves in time. The impact of maintenance of an element at a specific time on this behaviour is also reported in some of them. However optimization of maintenance activities (i.e., finding a minimum cost plan) is not considered which is the main motivation of our paper. Maintenance is expensive and critical in most systems. Unexpected breakdowns are not tolerable. That is why planning maintenance activities intelligently is an important issue since it saves money, service time and also lost production time. In this study, we are trying to optimize maintenance activities of a system where components age with a constant failure rate and there is a budget constraint. We develop a methodology to effectively prepare a predictive maintenance plan using DBNs. One can argue that the failure of the system and its associated costs can be modelled using influence diagrams or limited memory influence diagrams (LIMIDs). But we propose a way of representing the problem as an optimization problem first and then use only DBNs for fast inference under some simplifying assumptions.

The rest of the paper is organized as follows: In Section 2, problem is defined and in Section 3, dynamic Bayesian network based models are proposed as a solution to the problem defined. Two approaches are presented in scheduling maintenance plans. Numerical results are given in Section 4. Finally Section 5 gives conclusions and points future work.

## 2 Problem Definition

The problem we take up can be described as follows: There is a system which consists of several components. We observe the system in discrete epochs and assume that system reliability is observable. System reliability is a function of the interactions of the system components which are not directly observable. We presume that the reliability of the system should be kept over a predetermined threshold value in all periods. This is reasonable in mission critical systems where the failure of the system is a very low probability event due to built in redundancy and other structural properties. Therefore, we do not explicitly model the case where the system actually fails. Components age with a constant rate and it is possible to replace components in any period. Once replaced, the components will work at their full capacity. There is a given maintenance budget for each period, which the total replacement cost in that period cannot exceed. Our aim is to minimize total maintenance cost in a planning horizon such that reliability of the system never falls below the threshold and maintenance budget is not exceeded. Furthermore we make the following assumptions:

(i) Lifetime of any component in the system is exponentially distributed. That is failure rate

of any component is constant for all periods.

(ii) All other conditional probability distributions used in the representation are discrete.

(iii) All components and the system have two states ("1" is the working state, "0" is the failure state).

(iv) Components can be replaced at the beginning of each period. Once they are replaced, their working state probability (i.e., their reliability) become 1 in that period.

The problem can be expressed as a mathematical optimization problem.

The model parameters are:

$i$ : index of components
$t$ : index of time periods
$\lambda_i$ : failure rate of component $i$
$R_{i1}$ : initial reliability of component $i$
$c_{it}$ : cost of replacing component $i$ in period $t$
$B_t$ : available maintenance budget in period $t$
$L$ : threshold value of system reliability
$f(\cdot)$: function mapping component reliabilities $R_{it}$ to system reliability $R_{0t}$

The decision variables are:

$$X_{it} = \begin{cases} 1 & \text{if component } i \text{ is replaced in period } t \\ 0 & \text{otherwise} \end{cases}$$

$R_{it}$ : reliability of component i in period $t$
$R_{0t}$ : reliability of system in period $t$

The Predictive Maintenance (PM) model can be formulated mathematically as follows:

$$Z(PM) = \min \sum_{t=1}^{T} \sum_{i=1}^{n} c_{it} X_{it} \tag{1}$$

subject to

$$\sum_{i=1}^{n} c_{it} X_{it} \leq B_t, \ \forall t \tag{2}$$

$$R_{0t} \geq L, \ \forall t \tag{3}$$

$$R_{it} = (1 - X_{it}) e^{-\lambda_i} R_{i,t-1} + X_{it}, \ \forall i, t \tag{4}$$

$$R_{0t} = f(R_{1t}, R_{2t}, ...R_{nt}), \ \forall t \tag{5}$$

$$X_{it} \in \{0, 1\}, \ \forall i, t \tag{6}$$

$$0 \leq R_{it} \leq 1, \ \forall i, t \tag{7}$$

The objective function (1) aims to minimize the total component replacement cost. Constraint set (2) represents the budget constraints. Constraint set (3) guarantees that system reliability in each period should be greater than the given threshold value. Constraints in (4) ensure that if components are replaced their reliability becomes 1, otherwise it will decrease with corresponding failure rates. System reliability in each period is calculated by constraint (5). Finally constraints (6) and (7) define the bounds on decision variables. In general, solving this problem may be quite difficult. The difficulty lies in constraint sets (4) and (5). Constraint set (4) defines a non-linear relation of the decision variables whereas constraint set (5) is much more generic. In fact, system reliability at time $t$ can be a function of whole history of the system. It is this set of constraints and the implied relationships of constraint set (4) that we represent using a dynamic Bayesian network.

Further assumptions are imposed in order to simplify the above problem:

(v) Replacement costs of all components in any period are the same and they are all normalized at one.

(vi) Available budget in any period is normalized at one.

These assumptions indicate that in any period, only one replacement can be planned and the objective function we are trying to minimize becomes the total number of replacements in a planning horizon.

## 3 Proposed Solution

The mathematical problem may be solved analytically or numerically once the constraint set (5) is made explicit. However, as the causal relations (represented with constraint set (5) in the problem formulation) between the components and the system becomes more complex, it gets difficult to represent and solve it mathematically. We represent the constraint set (5) using dynamic Bayesian Networks (DBNs) (Murphy, 2002). A DBN is an extended Bayesian network (BN) which includes a temporal dimension. BNs are a widely used formalism for repre-

senting uncertain knowledge. The main features of the formalism are a graphical encoding of a set of conditional independence relations and a compact way of representing a joint probability distribution between random variables (Pearl, 1988). BNs have the power to represent causal relations between components and the system using conditional probability distributions. It is possible to analyse the process over a large planning horizon with DBNs.

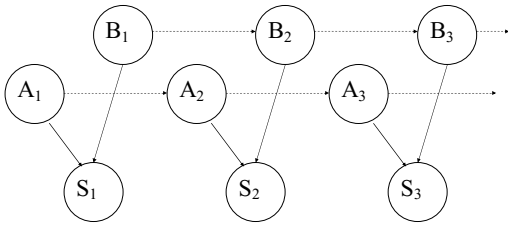



Figure 1: DBN representation of a system with 2 components

Figure 1 illustrates a DBN representation of a system with 2 components, A and B. Solid arcs represent the causal relations between the components and the system node whereas dashed arcs represent temporal relation of the components between two consecutive time periods. Note that this representation is Markovian whereas DBNs can represent more general transition structures. Temporal relations are the transition probabilities of components due to aging. Since the lifetime of any component in the system is exponentially distributed, the transition probabilities are constant because of the memoryless property of the exponential distribution given the time intervals are equal. Transition probability table for a component with two states ("1" is the working state, "0" is the failure state) is given in Table 1.

Table 1: Transition probability for component i

| | Comp($t$) | |
|---|---|---|
| Comp($t+1$) | 1 | 0 |
| 1 | $e^{-\lambda_i \triangle t}$ | 0 |
| 0 | 1-$e^{-\lambda_i \triangle t}$ | 1 |

DBN representation allows monitoring the system reliability in a given planning horizon and predicting the system state under different replacement schedules. When the system reliability falls below a predetermined threshold value, a component replacement is planned. The decision of which component to replace is an important issue since it affects future system reliability and consequently the next time to do a replacement. Like in decision-theoretic troubleshooting (Heckerman et al., 1995), marginal probabilities of components given the system state are used as efficiency measures of components in each period when a replacement is planned. Let $S_k$ the denote system state in period $k$ and $C_{ik}$ denote the state of component $i$ in period $k$. The following algorithm summarizes our DBN approach:

(i) Initialize $t = 1$
(ii) Infer system reliability $P(S_k = 1)$ $t \leq k \leq T$
(iii) Check if $P(S_k = 1) \leq L$
(iv) If $P(S_k = 1) \geq L$ $\forall k$, then stop.
(v) Else prepare a replacement plan for period $k$
  (a) Calculate $P_{ik} = P(C_{ik} = 0 | S_k = 0)$ $\forall i$
  (b) $i^* = \arg\max\{P_{ik}\}$
  (c) Update reliability of $i^*$ in $k$ to 1.
    $P(C_{ik} = 1) \leftarrow 1$
  (d) Update $t = k + 1$
(vi) If $t > T$, then stop.
(vii) Else continue with step (ii)

Note that $P(S_t = 1) = R_{0t}$ and $P(C_{it} = 1) = R_{it}$. This is a *myopic approach*, since the only evidence in calculating marginal probabilities in (v.a) belongs to the system state at the current planning time. An alternative approach is to take into account future information which can be transmitted by the transition probabilities of components. This is done by entering evidence to the system node from $k + 1$ to $T$ as $S_{k+1:T} = 0$. We call this approach the *look-ahead approach*. The algorithm is the same as above except for step (v.a) which is replaced as follows in look-ahead approach:

(v.a) Calculate $P_{ik} = P(C_{ik} = 0 | S_{k+1:T} = 0)$ $\forall i$ where $S_{k+1:T}$ denotes $S_{k+1}, ..., S_T$.

## 4 Numerical Results

The DBN algorithm is coded in Matlab and uses the Bayes Net Toolbox (BNT) (Murphy, 2001) to represent the causal and temporal relations, and to infer the reliability of the system. Two approaches, myopic and look-ahead, are compared on a small example with two components given in Figure 1. The planning horizon is taken as 100 periods and the threshold value is given as 0.50.

First scenario is created by taking mean time to failure (MTTF) $(1/\lambda_i)$ of each component $i$ equal which is set at 40 periods. The same replacement plan, given in Figure 2, is generated by both approaches. This is because components have equal MTTFs and hence equal transition probabilities. System reliability of scenario 1 is illustrated in Figure 2 where the peak points are the periods where a component is replaced. On each peak point, the component which is planned for replacement is indicated in the figure. 4 replacements are planned in both approaches. When a replacement occurs, system reliability jumps to a higher reliability value, and then gradually decreases as time evolves until the next replacement.

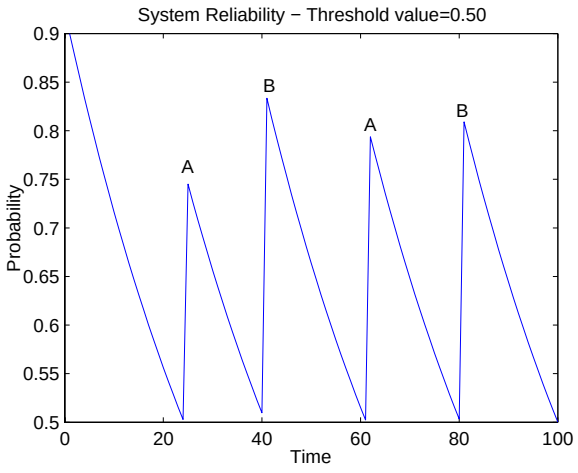



Figure 2: Scenario 1- system reliability and replacement plan

Second scenario is created by differentiating MTTFs of components. We decrease MTTF of component B to 10 periods. Replacement plans in Figure 3 and Figure 4 are generated by the myopic and look-ahead approaches, respectively. Replacement plans of the two approaches differ since components have different transition probabilities. Both approaches begin their replacement plan in period 12, by selecting the same component to replace. In the next replacement period ($k = 21$), different components are selected. Myopic approach selects component B, because this replacement will make the system reliability higher in the short-term. Look-ahead approach selects component A, because this replacement will make the system reliability higher in the long-run. This is further illustrated in Table 2. Although system reliability in the myopic approach is higher at $t = 21$, it decreases faster than the system reliability under the look-ahead approach. Hence, the look-ahead approach plans its next replacement at $t = 29$ while the myopic approach plans its next replacement at $t = 28$. By selecting A instead of B, the look-ahead approach defers its next replacement time. So as a total, in scenario 2, the look-ahead approach generates 10 replacements in 100 periods while the myopic approach generates 11.

Table 2: Scenario 2- system reliability where $21 \leq t \leq 28$

| Period | 21 | 22 | 23 | 24 |
|---|---|---|---|---|
| Myopic | .7712 | .7169 | .6673 | .6220 |
| Look-ahead | .7036 | .6718 | .6421 | .6144 |
| Period | 25 | 26 | 27 | 28 |
| Myopic | .5805 | .5425 | .5077 | .4758 |
| Look-ahead | .5884 | .5642 | .5414 | .5200 |

System reliability of scenario 2 is illustrated in Figures 3 and 4 for myopic and look-ahead approaches, respectively. In Figure 3, there are 11 peak points which means 11 replacements are planned. In Figure 4, there are 10 peak points which means 10 replacements are planned.

A third scenario is also carried out by further decreasing MTTF of component B to 5 periods. The discrepancy between plans generated by the two approaches becomes more apparent. Myopic approach plans a total of 19 replace-

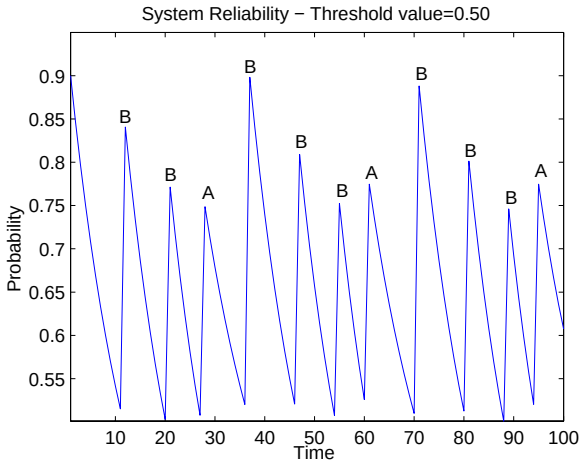



Figure 3: Scenario 2- system reliability and replacement plan with the myopic approach

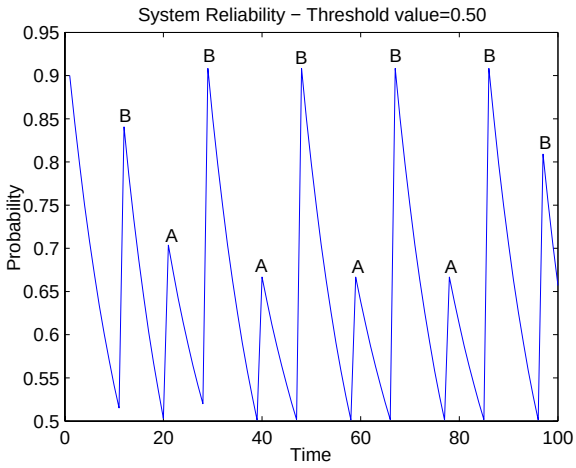



Figure 4: Scenario 2- system reliability and replacement plan with the look-ahead approach

ments while look-ahead approach plans a total of 16 replacements.

When the threshold value increases, an interesting question arises: Does it still worth to account for future information in choosing which component to replace? Table 3 shows the number of replacements found by the myopic and the look-ahead approaches at various threshold ($L$) values for scenario 2. When $L = 0.80$, myopic approach finds fewer replacements than the look-ahead approach. This is because as threshold increases, more frequent replacements will be planned, hence focusing on short-term reliability instead of the future reliability may result in less number of replacements.

Table 3: Number of replacements for the myopic and look-ahead approaches at various threshold values for $T = 100$

| Threshold | Myopic | Look-ahead |
|---|---|---|
| 0.50 | 11 | 10 |
| 0.60 | 15 | 15 |
| 0.70 | 23 | 22 |
| 0.80 | 33 | 35 |
| 0.90 | 67 | 67 |
| 0.95 | 100 | 100 |

In order to understand how good our methodology is, we enumerate all possible solutions of $k$ replacements in a reasonable time horizon. Here, $k$ refers to the upper bound of minimum replacements given by our DBN algorithm. The total number of solutions in a horizon of $T$ periods with $k$ replacements is given as:

$$\begin{pmatrix} T \\ k \end{pmatrix} 2^k \tag{8}$$

The first term is the total number of all possible size-$k$ subsets of a size-$T$ set. This corresponds to the total number of all possible time alternatives of $k$ replacements in horizon $T$. The second term is the total number of all possible replacements for two components. Since this solution space becomes intractable with increasing $T$ and $k$, a smaller part of the planning horizon is taken for enumeration of both scenarios. We started working with $T = 50$ periods where 2 repairs are proposed and $T = 30$ periods where 3 repairs are proposed by our algorithm for scenario 1 and scenario 2, respectively. The next replacements correspond to $t = 62$ (Figure 2) and $t = 40$ (Figure 4). Hence, we increased $T = 61$ and $T = 39$, the periods just before the 3rd and 4th replacements given by our algorithm for scenarios 1 and 2. The number of feasible solutions found by enumeration are reported in Table 4.

When we decrease $k$, number of replacements, by 1 ($k = 1$ and $k = 2$ for scenarios 1 and 2 re-

Table 4: Enumeration results

| Scenario | Horizon | Number of Replacements | Feasible Solutions |
|---|---|---|---|
| 1 | 50 | 2 | 324 |
| 1 | 50 | 1 | 0 |
| 1 | 61 | 2 | 2 |
| 1 | 61 | 1 | 0 |
| 2 | 30 | 3 | 1543 |
| 2 | 30 | 2 | 0 |
| 2 | 39 | 3 | 1 |
| 2 | 39 | 2 | 0 |

spectively); no feasible solution is found. So, it is not possible to find a plan with less replacements given by our algorithm for these cases. Note also that, in scenario 1 when $T = 61$ and $k = 2$, enumeration finds two feasible solutions which are in fact symmetric solutions found also by our algorithm. Similarly in scenario 2 when $T = 39$ and $k = 3$, enumeration finds one feasible solution which is the one found by our algorithm with the look-ahead approach. There are 1543 feasible solutions for scenario 2 with $T = 30$ and $k = 3$. Our lookahead approach finds one of these solutions and the solution it finds has the maximum system reliability at $T = 30$ among all solutions. The same observation is also valid for scenario 1 with $T = 50$ and $k = 2$.

## 5 Conclusion

We study dynamic reliability of a system where components age with a constant failure rate and there is a budget constraint. We develop a methodology to effectively prepare a good predictive maintenance plan using DBNs to represent the causal and temporal relations, and to infer reliability values. We try to minimize number of component replacements in a planning horizon by first deciding the time and then the component to replace. Two approaches are presented to choose the component and they are compared on three scenarios and various threshold values. When failure rates of components are equal, they find the same replacement plan. However, as failure rates differ, the two approaches may end up with different number of replacements. This is because the look-ahead approach takes future system reliability into consideration while the myopic approach focuses on the current planning time.

In this kind of predictive maintenance problem, there are two important decisions: One is the time of replacement. Replacement should be done such that system reliability is always guaranteed to be over a threshold. The other decision is which component(s) to replace in that period such that budget is not exceeded and total replacement cost is minimized. Our methodology is based on separating these decisions under assumptions (v) and (vi). We give the former decision by monitoring the first period when system reliability just falls below a given threshold. In other words, we defer a replacement decision as far as the threshold permits. As for the latter decision, we propose two approaches, myopic and look-ahead, to choose the component to replace. By enumerating feasible solutions in a reasonable horizon, we show that our method is effective for our simplified problem where the objective has become minimizing total replacements in a given planning horizon.

In this paper, we outline a method that can be used for finding a minimum-cost predictive maintenance schedule such that the system reliability is always above a certain threshold. The approach is normative in nature as opposed to descriptive which is the case in most of the literature that uses DBNs in reliability analysis.

The problem becomes more complex by (i) differentiating component costs (in time), (ii) differentiating available budget in time, (iii) defining a maintenance fixed cost for each period which may or may not differ in time. In these cases, separating the two decisions may not give a good solution. Studying such cases is left for future work.

**Acknowledgments**

This work is supported by Boğaziçi University Research Fund under grant BAP06A301D. Demet Özgür-Ünlüakın is supported by Boğaziçi University Foundation to attend the

PGM '06 workshop.

## References


Laurent Bouillaut, Philippe Weber, A. Ben Salem and Patrice Aknin. 2004. Use of causal probabilistic networks for the improvement of the maintenance of railway infrastructure. In *IEEE International Conference on Systems, Man and Cybernatics*, pages 6243-6249.

David Heckerman, John S. Breese and Koos Rommelse. 1995. Decision-theoretic troubleshooting, *Communications of the ACM*, 38(3): 49-57.

Finn V. Jensen, Uffe. Kjærulff, Brian. Kristiansen, Helge Langseth, Claus Skaanning, Jiri Vomlel and Marta Vomlelováá. 2001. The SACSO methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*,15(4):321-333.

Jayant Kalagnanam and Max Henrion. 1988. A comparison of decision analysis and expert rules for sequential analysis. In *4th Conference on Uncertainty in Artificial Intelligence*, pages 271-281.

Eylem Koca and Taner Bilgiç. 2004. Troubleshooting with dependent actions, conditional costs, and questions, Technical Report, FBE-IE-13/2004-18, Boğaziçi University.

Ranganath Kothamasu, Samuel H. Huang and William H. VerDuin. 2006. System health monitoring and prognostics - a review of current paradigms and practices. *Int J Adv Manuf Technol*, 28:1012-1024.

Helge Langseth and Finn V. Jensen. 2003. Decision theoretic troubleshooting of coherent systems. *Reliability Engineering and System Safety*, 80(1):49-62.

Helge Langseth and Finn V. Jensen. 2001. Heuristics for two extensions of basic troubleshooting. In *7th Scandinavian Conference on Artificial Intelligence, Frontiers in Artificial Intelligence and Applications*, pages 80-89.

Alexandre Muller, Philippe Weber and A. Ben Salem. 2004. Process model-based dynamic Bayesian networks for prognostic. In *IEEE 4th International Conference on Intelligent Systems Design and Applications.*

Kevin Patrick Murphy. 2002. Dynamic Bayesian networks: representation, inference and learning, *Ph.D. Dissertation*, University of California, Berkeley.

Kevin Patrick Murphy. 2001. The Bayes Net Toolbox for Matlab, *Computing Science and Statistics: Proceedings of the Interface.*

Jude Pearl. 1988. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*, Morgan Kaufmann Publishers.

Sampath Srinivas. 1995. A polynomial algorithm for computing the optimal repair strategy in a system with independent component failures. In *11th Annual Conference on Uncertainty in Artificial Intelligence*, pages 515-522.

Claus Skaanning, Finn V. Jensen and Uffe Kjærulff. 2000. Printer troubleshooting using Bayesian networks. In *13th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 367-379.

José Gerardo Torres-Toledano and Luis Enrique Sucar. 1998. Bayesian networks for reliability analysis of complex systems. In *6th Ibero-American Conference on AI: Progress in Artificial Intelligence*, pages 195-206.

W. E. Vesely. 1970. A time-dependent methodology for fault tree evaluation. *Nuclear Engineering and Design*, 13:337-360.

Philippe Weber and Lionel Jouffe. 2006. Complex system reliability modeling with Dynamic Object Oriented Bayesian Networks (DOOBN). *Reliability Engineering and System Safety*, 91: 149-162.

Philippe Weber, Paul Munteanu and Lionel Jouffe. 2004. Dynamic Bayesian networks modeling the dependability of systems with degradations and exogenous constraints. In *11th IFAC Symposium on Informational Control Problems in Manufacturing (INCOM'04).*

Philippe Weber and Lionel Jouffe. 2003. Reliability modeling with dynamic Bayesian networks. In *5th IFAC Symposium SAFEPROCESS'03*, pages 57-62.

Robert L. Welch and Travis V. Thelen. 2000. Dynamic reliability analysis in an operational context: the Bayesian network perspective. In *Dynamic Reliability: Future Directions*, pages 277-307.

# Reading Dependencies from the Minimal Undirected Independence Map of a Graphoid that Satisfies Weak Transitivity

Jose M. Peña
Linköping University
Linköping, Sweden

Roland Nilsson
Linköping University
Linköping, Sweden

Johan Björkegren
Karolinska Institutet
Stockholm, Sweden

Jesper Tegnér
Linköping University
Linköping, Sweden

**Abstract**

We present a sound and complete graphical criterion for reading dependencies from the minimal undirected independence map of a graphoid that satisfies weak transitivity. We argue that assuming weak transitivity is not too restrictive.

## 1 Introduction

A minimal undirected independence map $G$ of an independence model $p$ is used to read independencies that hold in $p$. Sometimes, however, $G$ can also be used to read dependencies holding in $p$. For instance, if $p$ is a graphoid that is faithful to $G$ then, by definition, vertex separation is a sound and complete graphical criterion for reading dependencies from $G$. If $p$ is simply a graphoid, then there also exists a sound and complete graphical criterion for reading dependencies from $G$ (Bouckaert, 1995).

In this paper, we introduce a sound and complete graphical criterion for reading dependencies from $G$ under the assumption that $p$ is a graphoid that satisfies weak transitivity. Our criterion allows reading more dependencies than the criterion in (Bouckaert, 1995) at the cost of assuming weak transitivity. We argue that this assumption is not too restrictive. Specifically, we show that there exist important families of probability distributions that are graphoids and satisfy weak transitivity.

The rest of the paper is organized as follows. In Section 5, we present our criterion for reading dependencies from $G$. As will become clear later, it is important to first prove that vertex separation is sound and complete for reading independencies from $G$. We do so in Section 4. Equally important is to show that assuming that $p$ satisfies weak transitivity is not too restrictive. We do so in Section 3. We start by reviewing some key concepts in Section 2 and close with some discussion in Section 6.

## 2 Preliminaries

The following definitions and results can be found in most books on probabilistic graphical models, e.g. (Pearl, 1988; Studený, 2005). Let $\mathbf{U}$ denote a set of random variables. Unless otherwise stated, all the independence models and graphs in this paper are defined over $\mathbf{U}$. Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{W}$ denote four mutually disjoint subsets of $\mathbf{U}$. An independence model $p$ is a set of independencies of the form $\mathbf{X}$ is independent of $\mathbf{Y}$ given $\mathbf{Z}$. We represent that an independency is in $p$ by $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ and that an independency is not in $p$ by $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$. An independence model is a graphoid when it satisfies the following five properties: Symmetry $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{Y} \perp\!\!\!\perp \mathbf{X}|\mathbf{Z}$, decomposition $\mathbf{X} \perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$, weak union $\mathbf{X} \perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW}$, contraction $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W}|\mathbf{Z} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{YW}|\mathbf{Z}$, and intersection $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W}|\mathbf{ZY} \Rightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{YW}|\mathbf{Z}$. Any strictly positive probability distribution is a graphoid.

Let $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ denote that $\mathbf{X}$ is separated from $\mathbf{Y}$ given $\mathbf{Z}$ in a graph $G$. Specifically, $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ holds when every path in $G$ between $\mathbf{X}$ and $\mathbf{Y}$ is blocked by $\mathbf{Z}$. If $G$ is an undirected graph (UG), then a path in $G$ between $\mathbf{X}$ and $\mathbf{Y}$ is blocked by $\mathbf{Z}$ when there exists some $Z \in \mathbf{Z}$ in the path. If $G$ is a directed and acyclic graph (DAG), then a path in $G$ between $\mathbf{X}$ and $\mathbf{Y}$ is blocked by $\mathbf{Z}$ when there exists a node $Z$

in the path such that either (i) $Z$ does not have two parents in the path and $Z \in \mathbf{Z}$, or (ii) $Z$ has two parents in the path and neither $Z$ nor any of its descendants in $G$ is in $\mathbf{Z}$. An independence model $p$ is faithful to an UG or DAG $G$ when $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ iff $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$. Any independence model that is faithful to some UG or DAG is a graphoid. An UG $G$ is an undirected independence map of an independence model $p$ when $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ if $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$. Moreover, $G$ is a minimal undirected independence (MUI) map of $p$ when removing any edge from $G$ makes it cease to be an independence map of $p$. A Markov boundary of $X \in \mathbf{U}$ in an independence model $p$ is any subset $MB(X)$ of $\mathbf{U} \setminus X$ such that (i) $X \perp\!\!\!\perp \mathbf{U} \setminus X \setminus MB(X)|MB(X)$, and (ii) no proper subset of $MB(X)$ satisfies (i). If $p$ is a graphoid, then (i) $MB(X)$ is unique for all $X$, (ii) the MUI map $G$ of $p$ is unique, and (iii) two nodes $X$ and $Y$ are adjacent in $G$ iff $X \in MB(Y)$ iff $Y \in MB(X)$ iff $X \not\perp\!\!\!\perp Y|\mathbf{U} \setminus (XY)$.

A Bayesian network (BN) is a pair $(G, \theta)$ where $G$ is a DAG and $\theta$ are parameters specifying a probability distribution for each $X \in \mathbf{U}$ given its parents in $G$, $p(X|Pa(X))$. The BN represents the probability distribution $p = \prod_{X \in \mathbf{U}} p(X|Pa(X))$. Then, $G$ is an independence map of a probability distribution $p$ iff $p$ can be represented by a BN with DAG $G$.

## 3 WT Graphoids

Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$. We call WT graphoid to any graphoid that satisfies weak transitivity $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}V \Rightarrow \mathbf{X} \perp\!\!\!\perp V|\mathbf{Z} \vee V \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ with $V \in \mathbf{U} \setminus (\mathbf{XYZ})$. We now argue that there exist important families of probability distributions that are WT graphoids and, thus, that WT graphoids are worth studying. For instance, any probability distribution that is Gaussian or faithful to some UG or DAG is a WT graphoid (Pearl, 1988; Studený, 2005). There also exist probability distributions that are WT graphoids although they are neither Gaussian nor faithful to any UG or DAG. For instance, it follows from the theorem below that the probability distribution that results from marginalizing some nodes out and instantiating some others in a probability distribution that is faithful to some DAG is a WT graphoid, although it may be neither Gaussian nor faithful to any UG or DAG.

**Theorem 1.** *Let $p$ be a probability distribution that is a WT graphoid and let $\mathbf{W} \subseteq \mathbf{U}$. Then, $p(\mathbf{U} \setminus \mathbf{W})$ is a WT graphoid. If $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ has the same independencies for all $\mathbf{w}$, then $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ for any $\mathbf{w}$ is a WT graphoid.*

*Proof.* Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U} \setminus \mathbf{W}$. Then, $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ in $p(\mathbf{U} \setminus \mathbf{W})$ iff $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ in $p$ and, thus, $p(\mathbf{U} \setminus \mathbf{W})$ satisfies the WT graphoid properties because $p$ satisfies them. If $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ has the same independencies for all $\mathbf{w}$ then, for any $\mathbf{w}$, $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ in $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ iff $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW}$ in $p$. Then, $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ for any $\mathbf{w}$ satisfies the WT graphoid properties because $p$ satisfies them. □

We now show that it is not too restrictive to assume in the theorem above that $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ has the same independencies for all $\mathbf{w}$, because there exist important families of probability distributions whose all or almost all the members satisfy such an assumption. For instance, if $p$ is a Gaussian probability distribution, then $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ has the same independencies for all $\mathbf{w}$, because the independencies in $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ only depend on the variance-covariance matrix of $p$ (Anderson, 1984). Let us now consider all the multinomial probability distributions for which a DAG $G$ is an independence map and denote them by $M(G)$. The following theorem, which is inspired by (Meek, 1995), proves that the probability of randomly drawing from $M(G)$ a probability distribution $p$ such that $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ does not have the same independencies for all $\mathbf{w}$ is zero.

**Theorem 2.** *The probability distributions $p$ in $M(G)$ for which there exists some $\mathbf{W} \subseteq \mathbf{U}$ such that $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ does not have the same independencies for all $\mathbf{w}$ have Lebesgue measure zero wrt $M(G)$.*

*Proof.* The proof basically proceeds in the same way as that of Theorem 7 in (Meek, 1995), so we refer the reader to that paper for more details.

Let $\mathbf{W} \subseteq \mathbf{U}$ and let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three disjoint subsets of $\mathbf{U} \setminus \mathbf{W}$. For a constraint such as $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ to be true in $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ but false in $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w}')$, two conditions must be met. First, $sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$ must not hold in $G$ and, second, the following equations must be satisfied: $p(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}, \mathbf{W} = \mathbf{w})p(\mathbf{Z} = \mathbf{z}, \mathbf{W} = \mathbf{w}) - p(\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}, \mathbf{W} = \mathbf{w})p(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}, \mathbf{W} = \mathbf{w}) = 0$ for all $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. Each equation is a polynomial in the BN parameters corresponding to $G$, because each term $p(\mathbf{V} = \mathbf{v})$ in the equations is the summation of products of BN parameters (Meek, 1995). Furthermore, each polynomial is non-trivial, i.e. not all the values of the BN parameters corresponding to $G$ are solutions to the polynomial. To see it, note that there exists a probability distribution $q$ in $M(G)$ that is faithful to $G$ (Meek, 1995) and, thus, that $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW}$ in $q$ because $sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$ does not hold in $G$. Then, by permuting the states of the random variables, we can transform the BN parameter values corresponding to $q$ into BN parameter values for $p$ so that the polynomial does not hold. Let $sol(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w})$ denote the set of solutions to the polynomial for $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. Then, $sol(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w})$ has Lebesgue measure zero wrt $\mathbb{R}^n$, where $n$ is the number of linearly independent BN parameters corresponding to $G$, because it consists of the solutions to a non-trivial polynomial (Okamoto, 1973). Let $sol = \bigcup_{\mathbf{X},\mathbf{Y},\mathbf{Z},\mathbf{W}} \bigcup_{\mathbf{w}} \bigcap_{\mathbf{x},\mathbf{y},\mathbf{z}} sol(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w})$ and recall from above that the outer-most union only involves those cases for which $sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$ does not hold in $G$. Then, $sol$ has Lebesgue measure zero wrt $\mathbb{R}^n$, because the finite union and intersection of sets of Lebesgue measure zero has Lebesgue measure zero too. Consequently, the probability distributions $p$ in $M(G)$ such that $p(\mathbf{U} \setminus \mathbf{W}|\mathbf{W} = \mathbf{w})$ does not have the same independencies for all $\mathbf{w}$ have Lebesgue measure zero wrt $\mathbb{R}^n$ because they are contained in $sol$. These probability distributions also have Lebesgue measure zero wrt $M(G)$, because $M(G)$ has positive Lebesgue measure wrt $\mathbb{R}^n$ (Meek, 1995). □

Finally, we argue in Section 6 that it is not unrealistic to assume that the probability distribution underlying the learning data in most projects on gene expression data analysis, one of the hottest areas of research nowadays, is a WT graphoid.

## 4 Reading Independencies

By definition, *sep* is sound for reading independencies from the MUI map $G$ of a WT graphoid $p$, i.e. it only identifies independencies in $p$. Now, we prove that *sep* in $G$ is also complete in the sense that it identifies all the independencies in $p$ that can be identified by studying $G$ alone. Specifically, we prove that there exist multinomial and Gaussian probability distributions that are faithful to $G$. Such probability distributions have all and only the independencies that *sep* identifies from $G$. Moreover, such probability distributions must be WT graphoids because *sep* satisfies the WT graphoid properties (Pearl, 1988). The fact that *sep* in $G$ is complete, in addition to being an important result in itself, is important for reading as many dependencies as possible from $G$ (see Section 5).

**Theorem 3.** *Let $G$ be an UG. There exist multinomial and Gaussian probability distributions that are faithful to $G$.*

*Proof.* We first prove the theorem for multinomial probability distributions. Create a copy $H$ of $G$ and, then, replace every edge $X - Y$ in $H$ by $X \rightarrow W_{XY} \leftarrow Y$ where $W_{XY} \notin \mathbf{U}$ is an auxiliary node. Let $\mathbf{W}$ denote all the auxiliary nodes created. Then, $H$ is a DAG over $\mathbf{UW}$. Moreover, for any three mutually disjoint subsets $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ of $\mathbf{U}$, $sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$ in $H$ iff $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ in $G$.

The probability distributions $p(\mathbf{U}, \mathbf{W})$ in $M(H)$ that are faithful to $H$ and satisfy that $p(\mathbf{U}|\mathbf{W} = \mathbf{w})$ has the same independencies for all $\mathbf{w}$ have positive Lebesgue measure wrt $M(H)$ because (i) $M(H)$ has positive Lebesgue measure wrt $\mathbb{R}^n$ (Meek, 1995), (ii) the probability distributions in $M(H)$ that are not faithful to $H$ have Lebesgue measure zero wrt $M(H)$ (Meek, 1995), (iii) the probability distributions $p(\mathbf{U}, \mathbf{W})$ in $M(H)$ such that $p(\mathbf{U}|\mathbf{W} = \mathbf{w})$ does not have the same independencies for all $\mathbf{w}$ have

Lebesgue measure zero wrt $M(H)$ by Theorem 2, and (iv) the union of the probability distributions in (ii) and (iii) has Lebesgue measure zero wrt $M(H)$ because the finite union of sets of Lebesgue measure zero has Lebesgue measure zero.

Let $p(\mathbf{U}, \mathbf{W})$ denote any probability distribution in $M(H)$ that is faithful to $H$ and satisfies that $p(\mathbf{U}|\mathbf{W} = \mathbf{w})$ has the same independencies for all $\mathbf{w}$. As proven in the paragraph above, such a probability distribution exists. Fix any $\mathbf{w}$ and let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$. Then, $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ in $p(\mathbf{U}|\mathbf{W} = \mathbf{w})$ iff $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW}$ in $p(\mathbf{U}, \mathbf{W})$ iff $sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$ in $H$ iff $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ in $G$. Then, $p(\mathbf{U}|\mathbf{W} = \mathbf{w})$ is faithful to $G$.

The proof for Gaussian probability distributions is analogous. In this case, $p(\mathbf{U}, \mathbf{W})$ is a Gaussian probability distribution and thus, for any $\mathbf{w}$, $p(\mathbf{U}|\mathbf{W} = \mathbf{w})$ is Gaussian too (Anderson, 1984). Theorem 2 is not needed in the proof because, as discussed in Section 3, any Gaussian probability distribution $p(\mathbf{U}, \mathbf{W})$ satisfies that $p(\mathbf{U}|\mathbf{W} = \mathbf{w})$ has the same independencies for all $\mathbf{w}$. □

The theorem above has previously been proven for multinomial probability distributions in (Geiger and Pearl, 1993), but the proof constrains the cardinality of $\mathbf{U}$. Our proof does not constraint the cardinality of $\mathbf{U}$ and applies not only to multinomial but also to Gaussian probability distributions. It has been proven in (Frydenberg, 1990) that *sep* in an UG $G$ is complete in the sense that it identifies all the independencies holding in every Gaussian probability distribution for which $G$ is an independence map. Our result is stronger because it proves the existence of a Gaussian probability distribution with exactly these independencies. We learned from one of the reviewers, whom we thank for it, that a rather different proof of the theorem above for Gaussian probability distributions is reported in (Lněnička, 2005).

The theorem above proves that *sep* in the MUI map $G$ of a WT graphoid $p$ is complete in the sense that it identifies all the independencies in $p$ that can be identified by studying $G$ alone. However, *sep* in $G$ is not complete if being complete is understood as being able to identify all the independencies in $p$. Actually, no sound criterion for reading independencies from $G$ alone is complete in the latter sense. An example follows.

**Example 1.** Let $p$ be a multinomial (Gaussian) probability distribution that is faithful to the DAG $X \rightarrow Z \leftarrow Y$. Such a probability distribution exists (Meek, 1995). Let $G$ denote the MUI map of $p$, namely the complete UG. Note that $p$ is not faithful to $G$. However, by Theorem 3, there exists a multinomial (Gaussian) probability distribution $q$ that is faithful to $G$. As discussed in Section 3, $p$ and $q$ are WT graphoids. Let us assume that we are dealing with $p$. Then, no sound criterion can conclude $X \perp\!\!\!\perp Y|\emptyset$ by just studying $G$ because this independency does not hold in $q$, and it is impossible to know whether we are dealing with $p$ or $q$ on the sole basis of $G$.

## 5 Reading Dependencies

In this section, we propose a sound and complete criterion for reading dependencies from the MUI map of a WT graphoid. We define the dependence base of an independence model $p$ as the set of all the dependencies $X \not\perp\!\!\!\perp Y|\mathbf{U} \setminus (XY)$ with $X, Y \in \mathbf{U}$. If $p$ is a WT graphoid, then additional dependencies in $p$ can be derived from its dependence base via the WT graphoid properties. For this purpose, we rephrase the WT graphoid properties as follows. Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{W}$ denote four mutually disjoint subsets of $\mathbf{U}$. Symmetry $\mathbf{Y} \not\perp\!\!\!\perp \mathbf{X}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$. Decomposition $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z}$. Weak union $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z}$. Contraction $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \vee \mathbf{X} \not\perp\!\!\!\perp \mathbf{W}|\mathbf{Z}$ is problematic for deriving new dependencies because it contains a disjunction in the right-hand side and, thus, it should be split into two properties: Contraction1 $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{W}|\mathbf{Z}$, and contraction2 $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{W}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW}$. Likewise, intersection $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \vee \mathbf{X} \not\perp\!\!\!\perp \mathbf{W}|\mathbf{ZY}$ gives rise to intersection1 $\mathbf{X} \not\perp\!\!\!\perp \mathbf{YW}|\mathbf{Z} \wedge \mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{ZW} \Rightarrow \mathbf{X} \not\perp\!\!\!\perp \mathbf{W}|\mathbf{ZY}$, and intersection2

$\mathbf{X} \not\perp \mathbf{YW}|\mathbf{Z} \wedge \mathbf{X} \perp \mathbf{W}|\mathbf{ZY} \Rightarrow \mathbf{X} \not\perp \mathbf{Y}|\mathbf{ZW}$. Note that intersection1 and intersection2 are equivalent and, thus, we refer to them simply as intersection. Finally, weak transitivity $\mathbf{X} \not\perp V|\mathbf{Z} \wedge V \not\perp \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp \mathbf{Y}|\mathbf{Z} \vee \mathbf{X} \not\perp \mathbf{Y}|\mathbf{Z}V$ with $V \in \mathbf{U} \setminus (\mathbf{XYZ})$ gives rise to weak transitivity1 $\mathbf{X} \not\perp V|\mathbf{Z} \wedge V \not\perp \mathbf{Y}|\mathbf{Z} \wedge \mathbf{X} \perp \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X} \not\perp \mathbf{Y}|\mathbf{Z}V$, and weak transitivity2 $\mathbf{X} \not\perp V|\mathbf{Z} \wedge V \not\perp \mathbf{Y}|\mathbf{Z} \wedge \mathbf{X} \perp \mathbf{Y}|\mathbf{Z}V \Rightarrow \mathbf{X} \not\perp \mathbf{Y}|\mathbf{Z}$. The independency in the left-hand side of any of the properties above holds if the corresponding *sep* statement holds in the MUI map $G$ of $p$. This is the best solution we can hope for because, as discussed in Section 4, *sep* in $G$ is sound and complete. Moreover, this solution does not require more information about $p$ than what it is available, because $G$ can be constructed from the dependence base of $p$. We call the WT graphoid closure of the dependence base of $p$ to the set of all the dependencies that are in the dependence base of $p$ plus those that can be derived from it by applying the WT graphoid properties.

We now introduce our criterion for reading dependencies from the MUI map of a WT graphoid. Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$. Then, $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ denotes that $\mathbf{X}$ is connected to $\mathbf{Y}$ given $\mathbf{Z}$ in an UG $G$. Specifically, $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ holds when there exist some $X_1 \in \mathbf{X}$ and $X_n \in \mathbf{Y}$ such that there exists exactly one path in $G$ between $X_1$ and $X_n$ that is not blocked by $(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$. Note that there may exist several unblocked paths in $G$ between $\mathbf{X}$ and $\mathbf{Y}$ but only one between $X_1$ and $X_n$. We now prove that *con* is sound for reading dependencies from the MUI map of a WT graphoid, i.e. it only identifies dependencies in the WT graphoid. Actually, it only identifies dependencies in the WT graphoid closure of the dependence base of $p$. Hereinafter, $X_{1:n}$ denotes a path $X_1, \ldots, X_n$ in an UG.

**Theorem 4.** *Let $p$ be a WT graphoid and $G$ its MUI map. Then, con in $G$ only identifies dependencies in the WT graphoid closure of the dependence base of $p$.*

*Proof.* We first prove that if $X_{1:n}$ is the only path in $G$ between $X_1$ and $X_n$ that is not blocked by $\mathbf{Y} \subseteq \mathbf{U} \setminus X_{1:n}$, then $X_1 \not\perp X_n|\mathbf{Y}$. We prove it by induction over $n$. We first prove it for $n = 2$. Let $\mathbf{W}$ denote all the nodes in $\mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y}$ that are not separated from $X_1$ given $X_2\mathbf{Y}$ in $G$. Since $X_1$ and $X_2$ are adjacent in $G$, $X_1 \not\perp X_2|\mathbf{U} \setminus X_{1:2}$ and, thus, $X_1\mathbf{W} \not\perp X_2(\mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y} \setminus \mathbf{W})|\mathbf{Y}$ due to weak union. This together with $sep(X_1\mathbf{W}, \mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y} \setminus \mathbf{W}|X_2\mathbf{Y})$, which follows from the definition of $\mathbf{W}$, implies $X_1\mathbf{W} \not\perp X_2|\mathbf{Y}$ due to contraction1. Note that if $\mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y} \setminus \mathbf{W} = \emptyset$, then $X_1\mathbf{W} \not\perp X_2(\mathbf{U} \setminus X_{1:2} \setminus \mathbf{Y} \setminus \mathbf{W})|\mathbf{Y}$ directly implies $X_1\mathbf{W} \not\perp X_2|\mathbf{Y}$. In any case, this independency together with $sep(\mathbf{W}, X_2|X_1\mathbf{Y})$, because otherwise there exist several unblocked paths in $G$ between $X_1$ and $X_2$ which contradicts the definition of $\mathbf{Y}$, implies $X_1 \not\perp X_2|\mathbf{Y}$ due to contraction1. Note that if $\mathbf{W} = \emptyset$, then $X_1\mathbf{W} \not\perp X_2|\mathbf{Y}$ directly implies $X_1 \not\perp X_2|\mathbf{Y}$. Let us assume as induction hypothesis that the statement that we are proving holds for all $n < m$. We now prove it for $n = m$. Since the paths $X_{1:2}$ and $X_{2:m}$ contain less than $m$ nodes and $\mathbf{Y}$ blocks all the other paths in $G$ between $X_1$ and $X_2$ and between $X_2$ and $X_m$, because otherwise there exist several unblocked paths in $G$ between $X_1$ and $X_m$ which contradicts the definition of $\mathbf{Y}$, then $X_1 \not\perp X_2|\mathbf{Y}$ and $X_2 \not\perp X_m|\mathbf{Y}$ due to the induction hypothesis. This together with $sep(X_1, X_m|\mathbf{Y}X_2)$, which follows from the definition of $X_{1:m}$ and $\mathbf{Y}$, implies $X_1 \not\perp X_m|\mathbf{Y}$ due to weak transitivity2.

Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$. If $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ holds in $G$, then there exist some $X_1 \in \mathbf{X}$ and $X_n \in \mathbf{Y}$ such that $X_1 \not\perp X_n|(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$ due to the paragraph above and, thus, $\mathbf{X} \not\perp \mathbf{Y}|\mathbf{Z}$ due to weak union. Then, every *con* statement in $G$ corresponds to a dependency in $p$. Moreover, this dependency must be in the WT graphoid closure of the dependence base of $p$, because we have only used in the proof the dependence base of $p$ and the WT graphoid properties. □

We now prove that *con* is complete for reading dependencies from the MUI map of a WT graphoid $p$, in the sense that it identifies all the dependencies in $p$ that follow from the information about $p$ that is available, namely the de-

pendence base of $p$ and the fact that $p$ is a WT graphoid.

**Theorem 5.** *Let $p$ be a WT graphoid and $G$ its MUI map. Then, con in $G$ identifies all the dependencies in the WT graphoid closure of the dependence base of $p$.*

*Proof.* It suffices to prove (i) that all the dependencies in the dependence base of $p$ are identified by *con* in $G$, and (ii) that *con* satisfies the WT graphoid properties. Since the first point is trivial, we only prove the second point. Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{W}$ denote four mutually disjoint subsets of $\mathbf{U}$.

- Symmetry $con(\mathbf{Y}, \mathbf{X}|\mathbf{Z}) \Rightarrow con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$. Trivial.

- Decomposition $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \Rightarrow con(\mathbf{X}, \mathbf{YW}|\mathbf{Z})$. Trivial if $\mathbf{W}$ contains no node in the path $X_{1:n}$ in the left-hand side. If $\mathbf{W}$ contains some node in $X_{1:n}$, then let $X_m$ denote the closest node to $X_1$ such that $X_m \in X_{1:n} \cap \mathbf{W}$. Then, the path $X_{1:m}$ satisfies the right-hand side because $(\mathbf{X} \setminus X_1)(\mathbf{YW} \setminus X_m)\mathbf{Z}$ blocks all the other paths in $G$ between $X_1$ and $X_m$, since $(\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_n)\mathbf{Z}$ blocks all the paths in $G$ between $X_1$ and $X_m$ except $X_{1:m}$, because otherwise there exist several unblocked paths in $G$ between $X_1$ and $X_n$, which contradicts the left-hand side.

- Weak union $con(\mathbf{X}, \mathbf{Y}|\mathbf{ZW}) \Rightarrow con(\mathbf{X}, \mathbf{YW}|\mathbf{Z})$. Trivial because $\mathbf{W}$ contains no node in the path $X_{1:n}$ in the left-hand side.

- Contraction1 $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW}) \Rightarrow con(\mathbf{X}, \mathbf{W}|\mathbf{Z})$. Since $\mathbf{ZW}$ blocks all the paths in $G$ between $\mathbf{X}$ and $\mathbf{Y}$, then (i) the path $X_{1:n}$ in the left-hand side must be between $\mathbf{X}$ and $\mathbf{W}$, and (ii) all the paths in $G$ between $X_1$ and $X_n$ that are blocked by $\mathbf{Y}$ are also blocked by $(\mathbf{W} \setminus X_n)\mathbf{Z}$ and, thus, $\mathbf{Y}$ is not needed to block all the paths in $G$ between $X_1$ and $X_n$ except $X_{1:n}$. Then, $X_{1:n}$ satisfies the right-hand side.

- Contraction2 $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{W}|\mathbf{Z}) \Rightarrow con(\mathbf{X}, \mathbf{Y}|\mathbf{ZW})$. Since $\mathbf{Z}$ blocks all the paths in $G$ between $\mathbf{X}$ and $\mathbf{W}$, the path $X_{1:n}$ in the left-hand side must be between $\mathbf{X}$ and $\mathbf{Y}$ and, thus, it satisfies the right-hand side.

- Intersection $con(\mathbf{X}, \mathbf{YW}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{Y}|\mathbf{ZW}) \Rightarrow con(\mathbf{X}, \mathbf{W}|\mathbf{ZY})$. Since $\mathbf{ZW}$ blocks all the paths in $G$ between $\mathbf{X}$ and $\mathbf{Y}$, the path $X_{1:n}$ in the left-hand side must be between $\mathbf{X}$ and $\mathbf{W}$ and, thus, it satisfies the right-hand side.

- Weak transitivity2 $con(\mathbf{X}, X_m|\mathbf{Z}) \wedge con(X_m, \mathbf{Y}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z}X_m) \Rightarrow con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ with $X_m \in \mathbf{U} \setminus (\mathbf{XYZ})$. Let $X_{1:m}$ and $X_{m:n}$ denote the paths in the first and second, respectively, *con* statements in the left-hand side. Let $X_{1:m:n}$ denote the path $X_1, \ldots, X_m, \ldots, X_n$. Then, $X_{1:m:n}$ satisfies the right-hand side because (i) $\mathbf{Z}$ does not block $X_{1:m:n}$, and (ii) $\mathbf{Z}$ blocks all the other paths in $G$ between $X_1$ and $X_n$, because otherwise there exist several unblocked paths in $G$ between $X_1$ and $X_m$ or between $X_m$ and $X_n$, which contradicts the left-hand side.

- Weak transitivity1 $con(\mathbf{X}, X_m|\mathbf{Z}) \wedge con(X_m, \mathbf{Y}|\mathbf{Z}) \wedge sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) \Rightarrow con(\mathbf{X}, \mathbf{Y}|\mathbf{Z}X_m)$ with $X_m \in \mathbf{U} \setminus (\mathbf{XYZ})$. This property never applies because, as seen in weak transitivity2, $sep(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ never holds since $\mathbf{Z}$ does not block $X_{1:m:n}$.

□

Note that the meaning of completeness in the theorem above differs from that in Theorem 3. It remains an open question whether *con* in $G$ identifies all the dependencies in $p$ that can be identified by studying $G$ alone. Note also that *con* in $G$ is not complete if being complete is understood as being able to identify all the dependencies in $p$. Actually, no sound criterion for reading dependencies from $G$ alone is complete in this sense. Example 1 illustrates this point. Let us now assume that we are dealing with $q$ instead of with $p$. Then, no sound criterion can

conclude $X \not\perp\!\!\!\perp Y|\emptyset$ by just studying $G$ because this dependency does not hold in $p$, and it is impossible to know whether we are dealing with $p$ or $q$ on the sole basis of $G$.

We have defined the dependence base of a WT graphoid $p$ as the set of all the dependencies $X \not\perp\!\!\!\perp Y|\mathbf{U} \setminus (XY)$ with $X, Y \in \mathbf{U}$. However, Theorems 4 and 5 remain valid if we redefine the dependence base of $p$ as the set of all the dependencies $X \not\perp\!\!\!\perp Y|MB(X) \setminus Y$ with $X, Y \in \mathbf{U}$. It suffices to prove that the WT graphoid closure is the same for both dependence bases of $p$. Specifically, we prove that the first dependence base is in the WT graphoid closure of the second dependence base and vice versa. If $X \not\perp\!\!\!\perp Y|\mathbf{U} \setminus (XY)$, then $X \not\perp\!\!\!\perp Y(\mathbf{U} \setminus (XY) \setminus (MB(X) \setminus Y))|MB(X) \setminus Y$ due to weak union. This together with $sep(X, \mathbf{U} \setminus (XY) \setminus (MB(X) \setminus Y)|Y(MB(X) \setminus Y))$ implies $X \not\perp\!\!\!\perp Y|MB(X) \setminus Y$ due to contraction1. On the other hand, if $X \not\perp\!\!\!\perp Y|MB(X) \setminus Y$, then $X \not\perp\!\!\!\perp Y(\mathbf{U} \setminus (XY) \setminus (MB(X) \setminus Y))|MB(X) \setminus Y$ due to decomposition. This together with $sep(X, \mathbf{U} \setminus (XY) \setminus (MB(X) \setminus Y)|Y(MB(X) \setminus Y))$ implies $X \not\perp\!\!\!\perp Y|\mathbf{U} \setminus (XY)$ due to intersection.

In (Bouckaert, 1995), the following sound and complete (in the same sense as *con*) criterion for reading dependencies from the MUI map of a graphoid is introduced: Let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three mutually disjoint subsets of $\mathbf{U}$, then $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$ when there exist some $X_1 \in \mathbf{X}$ and $X_2 \in \mathbf{Y}$ such that $X_1 \in MB(X_2)$ and either $MB(X_1) \setminus X_2 \subseteq (\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_2)\mathbf{Z}$ or $MB(X_2) \setminus X_1 \subseteq (\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_2)\mathbf{Z}$. Note that $con(\mathbf{X}, \mathbf{Y}|\mathbf{Z})$ coincides with this criterion when $n = 2$ and either $MB(X_1) \setminus X_2 \subseteq (\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_2)\mathbf{Z}$ or $MB(X_2) \setminus X_1 \subseteq (\mathbf{X} \setminus X_1)(\mathbf{Y} \setminus X_2)\mathbf{Z}$. Therefore, *con* allows reading more dependencies than the criterion in (Bouckaert, 1995) at the cost of assuming weak transitivity which, as discussed in Section 3, is not a too restrictive assumption.

Finally, the soundness of *con* allows us to give an alternative proof to the following theorem, which was originally proven in (Becker et al., 2000).

**Theorem 6.** *Let $p$ be a WT graphoid and $G$ its MUI map. If $G$ is a forest, then $p$ is faithful to it.*

*Proof.* Any independency in $p$ for which the corresponding separation statement does not hold in $G$ contradicts Theorem 4. □

## 6 Discussion

In this paper, we have introduced a sound and complete criterion for reading dependencies from the MUI map of a WT graphoid. In (Peña et al., 2006), we show how this helps to identify all the nodes that are relevant to compute all the conditional probability distributions for a given set of nodes without having to learn a BN first. We are currently working on a sound and complete criterion for reading dependencies from a minimal directed independence map of a WT graphoid.

Due to lack of time, we have not been able to address some of the questions posed by the reviewers. We plan to do it in an extended version of this paper. These questions were studying the relation between the new criterion and lack of vertex separation, studying the complexity of the new criterion, and studying the uniqueness and consistency of the WT graphoid closure.

Our end-goal is to apply the results in this paper to our project on atherosclerosis gene expression data analysis in order to learn dependencies between genes. We believe that it is not unrealistic to assume that the probability distribution underlying our data satisfies strict positivity and weak transitivity and, thus, it is a WT graphoid. The cell is the functional unit of all the organisms and includes all the information necessary to regulate its function. This information is encoded in the DNA of the cell, which is divided into a set of genes, each coding for one or more proteins. Proteins are required for practically all the functions in the cell. The amount of protein produced depends on the expression level of the coding gene which, in turn, depends on the amount of proteins produced by other genes. Therefore, a dynamic Bayesian network is a rather accurate model of the cell (Murphy and Mian, 1999): The nodes represent the genes and proteins, and the edges and parameters re-

present the causal relations between the gene expression levels and the protein amounts. It is important that the Bayesian network is dynamic because a gene can regulate some of its regulators and even itself with some time delay. Since the technology for measuring the state of the protein nodes is not widely available yet, the data in most projects on gene expression data analysis are a sample of the probability distribution represented by the dynamic Bayesian network after marginalizing the protein nodes out. The probability distribution with no node marginalized out is almost surely faithful to the dynamic Bayesian network (Meek, 1995) and, thus, it satisfies weak transitivity (see Section 3) and, thus, so does the probability distribution after marginalizing the protein nodes out (see Theorem 1). The assumption that the probability distribution sampled is strictly positive is justified because measuring the state of the gene nodes involves a series of complex wet-lab and computer-assisted steps that introduces noise in the measurements (Sebastiani et al., 2003).

Additional evidence supporting the claim that the results in this paper can be helpful for learning gene dependencies comes from the increasing attention that graphical Gaussian models of gene networks have been receiving from the bioinformatics community (Schäfer and Strimmer, 2005). A graphical Gaussian model of a gene network is not more than the MUI map of the probability distribution underlying the gene network, which is assumed to be Gaussian, hence the name of the model. Then, this underlying probability distribution is a WT graphoid and, thus, the results in this paper apply.

## Acknowledgments

We thank the anonymous referees for their comments. This work is funded by the Swedish Research Council (VR-621-2005-4202), the Ph.D. Programme in Medical Bioinformatics, the Swedish Foundation for Strategic Research, Clinical Gene Networks AB, and Linköping University.

## References

Theodore W. Anderson. 1984. *An Introduction to Multivariate Statistical Analysis*. John Wiley & Sons.

Ann Becker, Dan Geiger and Christopher Meek. 2000. Perfect Tree-Like Markovian Distributions. In *16th Conference on UAI*, pages 19–23.

Remco R. Bouckaert. 1995. *Bayesian Belief Networks: From Construction to Inference*. PhD Thesis, University of Utrecht.

Morten Frydenberg. 1990. Marginalization and Collapsability in Graphical Interaction Models. *Annals of Statistics*, 18:790–805.

Dan Geiger and Judea Pearl. 1993. Logical and Algorithmic Properties of Conditional Independence and Graphical Models. *The Annals of Statistics*, 21:2001–2021.

Radim Lněnička. 2005. On Gaussian Conditional Independence Structures. Technical Report 2005/ 14, Academy of Sciences of the Czech Republic.

Christopher Meek. 1995. Strong Completeness and Faithfulness in Bayesian Networks. In *11th Conference on UAI*, pages 411–418.

Kevin Murphy and Saira Mian. 1999. *Modelling Gene Expression Data Using Dynamic Bayesian Networks*. Technical Report, University of California.

Masashi Okamoto. 1973. Distinctness of the Eigenvalues of a Quadratic Form in a Multivariate Sample. *Annals of Statistics*, 1:763–765.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Jose M. Peña, Roland Nilsson, Johan Björkegren and Jesper Tegnér. 2006. Identifying the Relevant Nodes Without Learning the Model. In *22nd Conference on UAI*, pages 367–374.

Juliane Schäfer and Korbinian Strimmer. 2005. Learning Large-Scale Graphical Gaussian Models from Genomic Data. In *Science of Complex Networks: From Biology to the Internet and WWW*.

Paola Sebastiani, Emanuela Gussoni, Isaac S. Kohane and Marco F. Ramoni. 2003. Statistical Challenges in Functional Genomics (with Discussion). *Statistical Science*, 18:33–60.

Milan Studený. 2005. *Probabilistic Conditional Independence Structures*. Springer.

# Evidence and Scenario Sensitivities in Naive Bayesian Classifiers

Silja Renooij and Linda C. van der Gaag
Department of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{silja,linda}@cs.uu.nl

**Abstract**

Empirical evidence shows that naive Bayesian classifiers perform quite well compared to more sophisticated network classifiers, even in view of inaccuracies in their parameters. In this paper, we study the effects of such parameter inaccuracies by investigating the sensitivity functions of a naive Bayesian classifier. We demonstrate that, as a consequence of the classifier's independence properties, these sensitivity functions are highly constrained. We investigate whether the various patterns of sensitivity that follow from these functions support the observed robustness of naive Bayesian classifiers. In addition to the standard sensitivity given the available evidence, we also study the effect of parameter inaccuracies in view of scenarios of additional evidence. We show that the standard sensitivity functions suffice to describe such scenario sensitivities.

## 1 Introduction

Bayesian networks are often employed for classification tasks where an input instance is to be assigned to one of a number of output classes. The actual classifier then is a function which assigns a single class to each input, based on the posterior probability distribution computed from the Bayesian network for the output variable. Such classifiers are often built upon a naive Bayesian network, consisting of a single class variable and a number of observable feature variables, each of which is modelled as being independent of every other feature variable given the class variable. The numerical parameters for such a naive network are generally estimated from data and inevitably are inaccurate.

Experiments have shown time and again that classifiers built on naive Bayesian networks are quite stable: they are competitive with other learners, regardless of the size and quality of the data set from which they are learned (Domingos & Pazzani, 1997; Rish, 2001; Liu et al., 2005). Apparently, inaccuracies in the parameters of the underlying network do not significantly affect the performance of such a naive classifier.

The observed stability of naive Bayesian classifiers may be attributed to (a combination of) properties of the data and of the classifier function. The commonly used winner-takes-all rule, which assigns an instance to a class which is most probable according to the underlying Bayesian network, for example, seems to contribute to the naive classifier's success (Domingos & Pazzani, 1997). The observed stability may also be attributed to the naive independence properties of the classifier. It has been shown, for example, that naive Bayesian classifiers perform well for both completely independent and functionally dependent feature variables (Domingos & Pazzani, 1997; Rish, 2001).

In this paper, we employ techniques from sensitivity analysis to study the effects of parameter inaccuracies on a naive Bayesian network's posterior probability distributions and thereby contribute yet another explanation of the observed stability of naive Bayesian classifiers. In general, the sensitivity of a posterior probability of interest to parameter variation complies with a simple mathematical function (Castillo *et al.*, 1997; Coupé & Van der Gaag, 2002), called a sensitivity function. We will demonstrate that the independence assumptions underlying a naive Bayesian network constrain these sensi-

tivity functions to such a large extent that they can in fact be established exactly from very limited information. In addition, we study the various sensitivity properties that follow from the constrained functions. We would like to note that sensitivity analysis has been applied before in the context of naive Bayesian classifiers, as a means of providing bounds on the amount of parameter variation that is allowed without changing, for *any* of the possible instances, the class an instance is assigned to (Chan & Darwiche, 2003). We extend on this result with further insights.

For classification problems, it is often assumed that evidence is available for every single feature variable. In numerous application domains, however, this assumption may not be realistic. The question then arises how much impact additional evidence could have on the output of interest and how sensitive this impact is to inaccuracies in the network's parameters. We introduce the novel notion of scenario sensitivity to capture the latter type of sensitivity and show that the effects of parameter variation in view of scenarios of additional evidence can be established efficiently.

The paper is organised as follows. In Section 2, we present some preliminaries on sensitivity functions and their associated sensitivity properties. In Section 3, we establish the functional forms of the sensitivity functions for naive Bayesian networks and address the ensuing sensitivity properties. In Section 4 we introduce the notion of scenario sensitivity and show that it can be established from standard sensitivity functions. The paper ends with our conclusions and directions for future research in Section 5.

## 2 Preliminaries

To investigate the effects of inaccuracies in its parameters, a Bayesian network can be subjected to a sensitivity analysis. In such an analysis, the effects of varying a single parameter on an output probability of interest are studied. These effects are described by a simple mathematical function, called a *sensitivity function*. If, upon varying a single parameter, the parameters pertaining to the same conditional distribution are co-varied proportionally, then the sensitivity function is a quotient of two linear functions in the parameter $x$ under study (Castillo *et al.*, 1997; Coupé & Van der Gaag, 2002); more formally, the function takes the form

$$f(x) = \frac{a \cdot x + b}{c \cdot x + d}$$

where the constants $a, b, c, d$ are built from the assessments for the non-varied parameters. The constants of the function can be feasibly computed from the network (Coupé & Van der Gaag, 2002; Kjærulff & Van der Gaag, 2000).

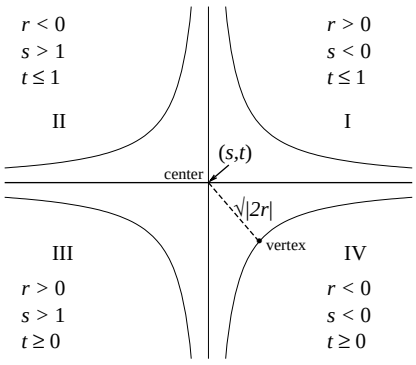



Figure 1: The possible hyperbolas and their constants, where the constraints on $r$, $s$ and $t$ are specific for sensitivity functions.

A sensitivity function is either a *linear* function or a fragment of a *rectangular hyperbola*. A rectangular hyperbola takes the general form

$$f(x) = \frac{r}{x - s} + t$$

where, for a function with $a, b, c, d$ as before, we have that $s = -\frac{d}{c}$, $t = \frac{a}{c}$, and $r = \frac{b}{c} + s \cdot t$. The hyperbola has two branches and the two asymptotes $x = s$ and $f(x) = t$; Figure 1 illustrates the locations of the possible branches relative to the asymptotes. Since both $x$ and $f(x)$ are probabilities, the two-dimensional space of their feasible values is defined by $0 \leq x, f(x) \leq 1$; this space is termed the *unit window*. Since any sensitivity function should be well-behaved in the unit window, a hyperbolic sensitivity function is a fragment of just one of the four possible branches in Figure 1.

From a sensitivity function, various properties can be computed. Here we briefly re-

view the properties of *sensitivity value* (Laskey, 1995), *vertex-proximity* and *admissible deviation* (Van der Gaag & Renooij, 2001). The *sensitivity value* for a parameter is the absolute value $\left|\frac{\partial f}{\partial x}(x_0)\right|$ of the first derivative of the sensitivity function at the assessment $x_0$ for the parameter; it captures the effect of an infinitesimally small shift in the parameter on the output probability. The impact of a larger shift is strongly dependent upon the location of the *vertex* of the sensitivity function, that is, the point where $\left|\frac{\partial f}{\partial x}(x)\right| = 1$. A vertex that lies within the unit window basically marks the transition from parameter values with a high sensitivity value to parameter values with a low sensitivity value, or vice versa. If an output probability is used for establishing the most likely value of the output variable, the effect of parameter variation on the most likely output value is of interest. The *admissible deviation* for a parameter now is a pair $(\alpha, \beta)$, where $\alpha$ is the amount of variation allowed to values smaller than its assessment without changing the most likely output value, and $\beta$ is the amount of variation allowed to larger values; the symbols $\leftarrow$ and $\rightarrow$ are used to indicate that variation is allowed up to the boundaries of the unit window.

## 3 Sensitivities in Naive Classifiers

Upon being subjected to a sensitivity analysis, the independence assumptions of a naive Bayesian network strongly constrain the general form of the resulting sensitivity functions. In fact, given just limited information, the exact functions can be readily determined for each class value and each parameter. In this section we derive these functions and discuss their ensuing sensitivity properties.

### 3.1 Functional forms

Before establishing the sensitivity functions for naive Bayesian networks, we introduce some notational conventions. We consider a naive Bayesian network with nodes $V(G) = \{C\} \cup E$, $E = \{E_1, \ldots, E_n\}$, $n \geq 2$, and arcs $A(G) = \{(C, E_i) \mid i = 1, \ldots, n\}$; $C$ is called the *class variable* of the network and $E_i$ are termed its *feature variables*. We now study the sensitivity of the posterior probability $\Pr(c \mid e)$ of a class value $c$ given an input instance $e$ in terms of a parameter $x = p(e'_v \mid c')$, where $e'_v$ denotes some value of the feature variable $E_v$ and $c'$ is a value of the class variable. The original assessment specified in the network for the parameter $x$ is denoted by $x_0$. The notation $p_c$ is used for the posterior probability of interest prior to parameter variation. We use $f_{\Pr(c|e)}(x)$ to denote the sensitivity function that expresses the probability of interest in terms of the parameter $x$.

The following proposition now captures the sensitivity function that describes the output probability of interest of a naive Bayesian network in terms of a single parameter associated with one of the feature variables. The proposition shows that this function is highly constrained as a result of the independences in the network and the parameter being conditioned on a value of the class variable. In fact, for any class value and any parameter, only one of four different functions can result. We would like to note that the proposition holds only for parameters and output probabilities that allow for a hyperbolic sensitivity function.

**Proposition 1.** *Let $E_v$ be a feature variable with the value $e_v$ and let $x = p(e'_v \mid c')$ be a parameter associated with $E_v$. Then, the sensitivity function $f_{\Pr(c|e)}(x)$ has one of the following forms:*

| $f_{\Pr(c\mid e)}(x)$ | $e_v = e'_v$ | $e_v \neq e'_v$ |
|---|---|---|
| $c = c'$ | $\dfrac{x}{x - s_1}$ | $\dfrac{x-1}{x - s_2}$ |
| $c \neq c'$ | $p_c \cdot \dfrac{x_0 - s_1}{x - s_1}$ | $p_c \cdot \dfrac{x_0 - s_2}{x - s_2}$ |

*where $s_1 = x_0 - \frac{x_0}{p'}$ and $s_2 = x_0 + \frac{(1-x_0)}{p'}$ are the vertical asymptotes of the corresponding functions and $p'$ is the original value of $\Pr(c' \mid e)$.*

*Proof.* We prove the property stated in the proposition for $e_v = e'_v$; the proof for $e_v \neq e'_v$ is analogous. We begin by writing the probability $\Pr(c, e)$ in terms of the network's parameters:

$$\Pr(c, e) = \prod_{E_i \in E \setminus \{E_v\}} p(e_i \mid c) \cdot p(c) \cdot p(e_v \mid c),$$

where $e_i$ is the value of the variable $E_i$ in the input instance $e$. This probability relates to the parameter $x = p(e'_v \mid c')$ as

$$\Pr(c', e)(x) = \prod_{E_i \in E \setminus \{E_v\}} p(e_i \mid c') \cdot p(c') \cdot x$$

for $c = c'$ and as

$$\Pr(c, e)(x) = \prod_{E_i \in E} p(e_i \mid c) \cdot p(c)$$

for $c \neq c'$. Similarly, $\Pr(e)$ can be written as

$$\Pr(e) = \Pr(c', e) + \sum_{c \neq c'} \Pr(c, e),$$

where $\Pr(c', e)$ again relates to the parameter $x$ as indicated above and the other term is constant with respect to the parameter. For $c = c'$ we now find for the sensitivity function that

$$f_{\Pr(c' \mid e)}(x) = \frac{\Pr(c', e)(x)}{\Pr(e)(x)} = \frac{a \cdot x}{a \cdot x + d} = \frac{x}{x - s_1}$$

and for $c \neq c'$ that

$$f_{\Pr(c \mid e)}(x) = \frac{\Pr(c, e)(x)}{\Pr(e)(x)} = \frac{b_c}{a \cdot x + d} = \frac{r_c}{x - s_1}$$

with the constants

$$\begin{aligned} a &= \prod_{E_i \in E \setminus \{E_v\}} p(e_i \mid c') \cdot p(c') \\ b_c &= \prod_{E_i \in E} p(e_i \mid c) \cdot p(c) \\ d &= \sum_{c \neq c'} \Pr(c, e) \end{aligned}$$

The constant $s_1 = -\frac{d}{a}$ is the vertical asymptote that is shared by all functions; its value is determined from

$$\begin{aligned} s_1 &= -\frac{\Pr(e) - \Pr(c', e)}{\Pr(c', e)/p(e_v \mid c')} \\ &= x_0 - \frac{x_0}{p'} \end{aligned}$$

In addition, the function $f_{\Pr(c' \mid e)}(x)$ has a horizontal asymptote at $t = \frac{a}{a} = 1$. The functions $f_{\Pr(c \mid e)}(x)$ with $c \neq c'$, all have $t = \frac{0}{a} = 0$. The constants $r_c = \frac{b_c}{a}$ of the latter functions are determined from $p_c = \frac{r_c}{x_0 - s_1}$. □

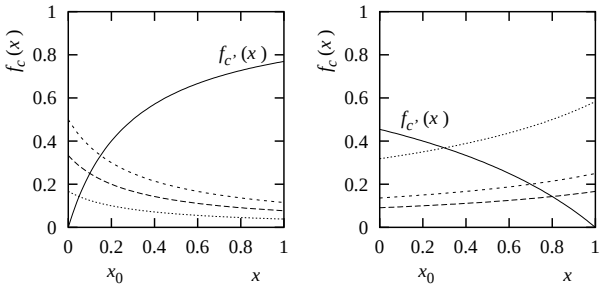



Figure 2: Example sensitivity functions for the parameter $x = p(e'_v \mid c')$ with $x_0 = 0.2$, $p' = 0.4$, where $e_v = e'_v$ (*left*) or $e_v \neq e'_v$ (*right*).

From the values of the asymptotes found in the proof above, we observe that for $e_v = e'_v$ the sensitivity function $f_{\Pr(c' \mid e)}(x)$ is a fragment of a IVth-quadrant hyperbola branch; all other functions $f_{\Pr(c \mid e)}(x)$ with $c \neq c'$ are fragments of Ist-quadrant branches. For $e_v \neq e'_v$ we find IIIrd- and IInd-quadrant branches, respectively. These properties are illustrated in Figure 2. To intuitively explain why the function $f_{\Pr(c' \mid e)}(x)$ has a different shape than the other functions, we observe that varying a parameter $p(e_i \mid c')$ has a direct effect on the probability $\Pr(c' \mid e)$, whereas the probabilities $\Pr(c \mid e)$, $c \neq c'$, are only indirectly affected to ensure that the distribution sums to one. Due to the constrained form of these functions, moreover, all $f_{\Pr(c \mid e)}(x)$, $c \neq c'$, have the same shape. The function $f_{\Pr(c' \mid e)}(x)$ therefore *must* be deviant.

We illustrate the functional forms derived above with an example. The example shows more specifically that as a result of their constrained form, any sensitivity function can be established from very limited information.

**Example 1.** We consider a naive Bayesian network with a class variable modelling the stages I, IIA, IIB, III, IVA and IVB of cancer of the oesophagus; the feature variables of the network capture the results from diagnostic tests. For a given patient, the available findings are summarised in the input instance $e$, given which the following posterior distribution is computed over the class variable $S$:

| $S$ | I | IIA | IIB | III | IVA | IVB |
|---|---|---|---|---|---|---|
| $\Pr(S \mid e)$ | 0.01 | 0.19 | 0.01 | 0.07 | **0.61** | 0.11 |

We further consider the feature variable *CT-loco*, modelling the observed presence or absence of loco-regional lymphatic metastases. The network includes the following assessments for the parameters for this variable:

| $S$ | I | IIA | IIB | III | IVA | IVB |
|---|---|---|---|---|---|---|
| *CT-loco* yes | 0.02 | 0.02 | 0.48 | 0.48 | 0.48 | 0.27 |
| no | 0.98 | 0.98 | 0.52 | 0.52 | **0.52** | 0.73 |

Now suppose that we are interested in the effects of inaccuracies in the parameter $x = p(CT\text{-}loco = \text{no} \mid S = \text{IVA})$ on the posterior probabilities $\Pr(S \mid e)$ for our patient who has no evidence of loco-regional metastases. These effects are captured by six functions with the vertical asymptote $s_1 = 0.52 - \frac{0.52}{0.61} = -0.33$. Without having to perform any further computations, we now find that

$$f_{\text{IVA}}(x) = \frac{x}{x + 0.33}$$

and

$$f_S(x) = \Pr(S \mid e) \cdot \frac{0.85}{x + 0.33}$$

for any $S \neq \text{IVA}$. We further find that for the complement of the parameter $x$ all functions have their vertical asymptote at $s_2 = 1.33$. □

### 3.2 Sensitivity properties

For a naive Bayesian network, any sensitivity function is exactly determined by the assessment for the parameter being varied and the original value of the probability of interest. From the function now any sensitivity property of interest can be computed. We study the properties of sensitivity value, vertex proximity and admissible deviation.

**Sensitivity value and vertex proximity** For the various possible sensitivity functions for a naive Bayesian network, the sensitivity values are readily established: for $e_v = e'_v$ we find that

$$\left|\frac{\partial f_{c'}}{\partial x}(x_0)\right| = (1 - p') \cdot \frac{p'}{x_0} \geq p_c \cdot \frac{p'}{x_0} = \left|\frac{\partial f_c}{\partial x}(x_0)\right|.$$

For $e_v \neq e'_v$, we find a similar result by replacing $x_0$ by $1 - x_0$. Note that for a binary class variable, the sensitivity values for its two class values are the same. This property holds for any binary output variable in any Bayesian network.

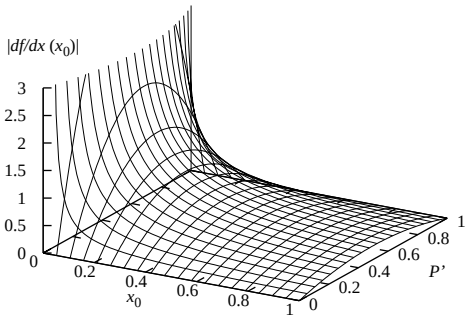



Figure 3: The sensitivity value for $f_{\Pr(c'|e)}(x)$ as a function of $x_0$ and $p'$, for $e_v = e'_v$.

Based upon the general form of the sensitivity value derived above, Figure 3 depicts the sensitivity value of $f_{\Pr(c'|e)}(x)$ given $e_v = e'_v$ for different combinations of values for $x_0$ and $p'$. We observe that high sensitivity values can only be found if the assessment for the parameter under study is small and the original posterior probability $p'$ is non-extreme. More formally, we have that $\left|\frac{\partial f_{\Pr(c'|e)}}{\partial x}(x_0)\right| > 1$ if and only if $x_0 < p' - p'^2 \leq 0.25$. For $e_v \neq e'_v$, high sensitivity values can only be found if the assessment for the parameter is larger than 0.75 and the original posterior probability $p'$ is non-extreme. Note that high sensitivity values can thus only be found for the parameters that describe the *least likely* value of a feature variable given a particular class value. This property was noticed before for general Bayesian networks (Van der Gaag & Renooij, 2006).

In view of naive Bayesian classifiers, we observe that input instances which occur the more frequently given a particular class are also likely to be the more probable given that class in the network underlying the classifier. Since high sensitivity values can only be found for the parameters that describe the least likely value of a feature variable, the posterior probability of the corresponding class value for such an input instance will not be very sensitive to inaccuracies in the various parameters. In fact, the vertical asymptote of an associated sensitivity function, and hence the $x$-value of the function's vertex, will be quite distant from the parameter's assessment, resulting in a rather flat function in

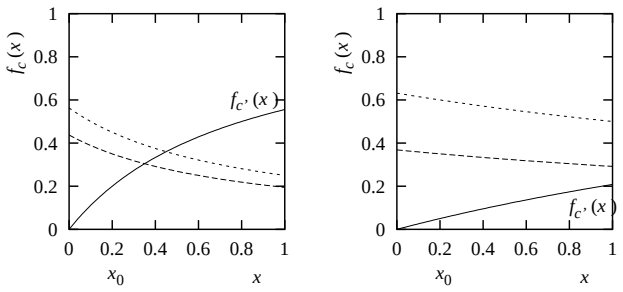



Figure 4: Examples of sensitivity functions for the parameter $x = p(e'_v \mid c')$, where $e_v = e'_v$ and $c'$ is *not* the most likely value. Variation either can make $c'$ the most likely value (*left*) or will not change the most likely value (*right*).

the broader vicinity of $x_0$. The most likely class value will then hardly ever change upon small shifts in a parameter. The above argument thus corroborates the empirically observed robustness of classifiers to parameter inaccuracies.

**Admissible deviation** In view of classification, the property of admissible deviation is especially of interest. The admissible deviation for a parameter gives the amount of variation that is allowed before an instance is classified as belonging to a different class. The following proposition now gives a general form for all possible deviations in a naive Bayesian network. The proposition more specifically shows that in such a network, the most likely class value can change *at most once* upon varying a parameter.

**Proposition 2.** *Let* $p^\top = \max\{p_c \mid c \neq c'\}$, *let* $c^\top$ *be a value of* $C$ *for which* $\Pr(c^\top \mid e) = p^\top$, *and let* $f^\top(x) = f_{\Pr(c^\top|e)}(x)$. *Then,*

- *we have* $f_{\Pr(c|e)}(x) \leq f^\top(x)$ *for all* $c \neq c'$;
- *for* $e_v = e'_v$ *the admissible deviation for the parameter* $x$ *is:*

| *condition* | *admissible deviation* | |
|---|---|---|
| $p^\top < p'$ | $(x_0 - x_s, \rightarrow)$ | |
| $p^\top = p'$ | $(0, 0)$ | |
| $p^\top > p'$ | $(\leftarrow, x_s - x_0)$ | *if* $x_0 < \frac{p'}{p^\top}$ |
| | $(\leftarrow, \rightarrow)$ | *otherwise* |

*where* $x_s = p^\top \cdot \frac{x_0}{p'}$;

- *for* $e_v \neq e'_v$ *then the above admissible deviations hold with each occurrence of* $x_0$ *and* $x_s$ *replaced by* $1-x_0$ *and* $1-x_s$, *respectively.*

*Proof.* The first property follows from the observation that all functions $f_{\Pr(c|e)}$ with $c \neq c'$ have the same horizontal and vertical asymptotes and therefore do not intersect. As a consequence, either $c'$ or $c^\top$ is the most likely value of $C$, regardless of the value of $x$. If the two functions $f_{\Pr(c'|e)}(x)$ and $f^\top(x)$ intersect at $x_s$, then we have that $x_s \in [0, \infty\rangle$ for $e_v = e'_v$ and $x_s \in \langle -\infty, 1]$ for $e_v \neq e'_v$. The second and third properties stated above now follow immediately from the general forms of the functions. □

We observe from the above proposition that, for any parameter, an admissible amount of variation different from $\leftarrow$ or $\rightarrow$ is possible only to the left or to the right of the parameter's assessment but never in both directions. From this observation we have that upon varying the parameter, the most likely class value can change only once. Figures 2 and 4 support this observation.

**Example 2.** We consider again the naive Bayesian classifier and the patient information from Example 1. Suppose that we are once more interested in the effects of inaccuracies in the parameter $x = p(\mathit{CT\text{-}loco} = \text{no} \mid S = \text{IVA})$ on the posterior probabilities $\Pr(S \mid e)$ for the patient. We recall that the value IVA corresponds to the most likely stage for this patient, with a probability of 0.61. The second most likely stage for the patient is stage IIA, with probability 0.19. We now find that the most likely stage changes from IVA to IIA at $x_s = 0.19 \cdot \frac{0.52}{0.61} = 0.16$. The admissible deviation for the parameter thus is $(0.36, \rightarrow)$. Note that the most likely stage cannot change into any other value upon varying the parameter under study. □

In view of naive Bayesian network classifiers, we have from the above proposition that we can expect to find large admissible deviations for mist instances. To support this expectation, we consider a parameter $x$ with $e_v = e'_v$; similar arguments hold for $e_v \neq e'_v$. If $p^\top < p'$, we have from the functional forms that the assessment $x_0$ for the parameter is unlikely to be very small, as for

functions with an asymptote to the left of the unit window the intersection of the sensitivity functions lies to the left of $x_0$. If $p^\top > p'$, on the other hand, we expect an admissible deviation within the unit window *only* for very small values of $x_0$. If the instances that are modelled as the more probable given a particular class value, occur the more often in practice, then for these instances we expect that $x_0$ is not a small value and that $p' > p^\top$. In practice we would therefore expect most instances to result in admissible deviations of $(\leftarrow, \rightarrow)$.

## 4 Scenario sensitivity

For classification problems, it is generally assumed that evidence is available for every single feature variable. In practical applications, however, this assumption may not be realistic. In the medical domain, for example, a patient is to be classified into one of a number of diseases without being subjected to every possible diagnostic test. The question then arises how much impact additional evidence could have on the probability distribution over the class variable and how sensitive this impact is to inaccuracies in the network's parameters. The former issue is closely related to the notion of value of information. The latter issue involves a notion of sensitivity that differs from the standard notion used in the previous sections in the sense that it pertains not to available evidence but to scenarios of possibly additional evidence. We refer to this notion of sensitivity as *scenario sensitivity* and use the term *evidence sensitivity* to refer to the more standard notion. Although it is applicable to Bayesian networks in general, in this section we discuss scenario sensitivity in the context of naive Bayesian classifiers.

To study the effect of additional evidence on an output probability for the class variable, we consider the ratio $\Pr(c \mid e^N)$ to $\Pr(c \mid e^O)$ where $e^O$ and $e^N$ denote the evidence prior to and after receiving the new evidence respectively.

**Proposition 3.** *Let $E^O$ and $E^N$ be sets of feature variables with $\varnothing \subseteq E^O \subset E^N \subseteq E$ and $E^N - E^O = \{E_1, \ldots, E_l\}$, $1 \leq l \leq n$. Let $e^O$ and $e^N$ be consistent instances of $E^O$ and $E^N$, respectively. Then, for each $c$, we have that*

$$\frac{\Pr(c \mid e^N)}{\Pr(c \mid e^O)} = \frac{\prod_{i=1}^{l} \Pr(e_i \mid c)}{\sum_{c_j} \prod_{i=1}^{l} \Pr(e_i \mid c_j) \cdot \Pr(c_j \mid e^O)}$$

*where $e_i$ is the value of $E_i$ in $E^N$.*

*Proof.* The property follows immediately by applying Bayes' rule and exploiting the independences that hold in the naive network. □

The above proposition now allows us to compute the new posterior probability distribution over the class variable from the previous one.

**Example 3.** We reconsider the naive Bayesian network and the patient information from the previous examples. In addition to the tests to which the patient has already been subjected, a scan of the liver can be performed. We now are interested in the posterior distribution over the various stages if the result of this test were positive. For the feature variable *CT-liver*, the following parameter assessments are specified:

| $S$ | I | IIA | IIB | III | IVA | IVB |
|---|---|---|---|---|---|---|
| *CT-liver* yes | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | **0.69** |
| no | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.31 |

From the table and the probability distribution $\Pr(S \mid e)$ from Example 1, we find that $\sum_S \Pr(\textit{CT-liver} = \text{yes} \mid S) \cdot \Pr(S \mid e) = 0.1204$. The new posterior probability of $S = \text{IVB}$ now follows directly from

$$\frac{\Pr(\text{IVB} \mid e^N)}{0.11} = \frac{0.69}{0.1204} = 0.63$$

without requiring any additional computations from the network. Note that the new test result would change the most likely stage. □

The above proposition allows for establishing the impact of additional evidence on the posterior probability distribution over the class variable. To capture the sensitivity of this impact to parameter variation, we consider the function $h(x)$ that describes the above probability ratio as a function of a parameter $x = p(e'_v \mid c')$:

$$h(x) = \left(\frac{\Pr(c \mid e^N)}{\Pr(c \mid e^O)}\right)(x) = \frac{\Pr(c \mid e^N)(x)}{\Pr(c \mid e^O)(x)}$$

If the parameter $x$ pertains to a variable in the set $E^N - E^O$, then the denominator is a constant with respect to $x$. The function $h(x)$ then

just is a *scaled* version of the sensitivity function $f_{\Pr(c|e^N)}(x)$. Given the probability distribution $\Pr(C \mid e^O)$ over the class variable, we can therefore immediately determine the sensitivity of the impact of the additional evidence from the sensitivity function $f_{\Pr(c|e^N)}(x)$. Note that in a naive Bayesian network the latter function is known for each parameter $x$ once the posterior probability distribution $\Pr(C \mid e^N)$ is available.

**Example 4.** We reconsider the previous example. We now are interested in the effects of inaccuracies in the parameter $x = p(CT\text{-}liver = \text{yes} \mid \text{IVB})$ on the ratio $\Pr(\text{IVB} \mid e^N)$ to $\Pr(\text{IVB} \mid e^O)$. We recall that the new posterior probability of $S = \text{IVB}$ would be 0.63; the probability given just the available evidence was 0.11. We now establish the sensitivity function $f_{\Pr(\text{IVB}|e^N)}(x) = \frac{x}{x+0.41}$ and find that

$$h_{\text{IVB}}(x) = \frac{1}{0.11} \cdot \frac{x}{x + 0.41}$$

From Example 3 we had that the probability of the class value IVB increased from 0.11 to 0.63 upon a positive liver scan, thereby becoming 5.7 times as likely. We can now in addition conclude that if the parameter $x$ is varied, the class value IVB can become at most 6.4 times as likely as without the additional evidence. □

## 5 Conclusions

In this paper, we used techniques from sensitivity analysis to study the effects of parameter inaccuracies on a naive Bayesian network's posterior probability distributions. We showed that the independence assumptions of such a network constrain the functional form of the associated sensitivity functions: these functions are determined solely by the assessment for the parameter under study and the original posterior probability distribution over the class variable. The sensitivity properties following from the functions provided some fundamental arguments which corroborated the empirically observed robustness of classifiers to parameter inaccuracies. More research is required, however, to further substantiate these arguments. We further introduced the novel notion of scenario sensitivity, which describes the effects of parameter inaccuracies in view of scenarios of additional evidence. We showed that for naive Bayesian networks scenario sensitivity can be readily expressed in terms of the more standard sensitivity functions. In the near future, we would like to study scenario sensitivity in Bayesian networks in general.

## References


E. Castillo, J.M. Gutiérrez and A.S. Hadi. 1997. Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 27: 412 – 423.

H. Chan and A. Darwiche. 2003. Reasoning about Bayesian network classifiers. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pages 107 – 115.

V.M.H. Coupé and L.C. van der Gaag. 2002. Properties of sensitivity analysis of Bayesian belief networks. *Annals of Mathematics and Artificial Intelligence*, 36: 323 – 356.

P. Domingos and M. Pazzani. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29: 103 – 130.

U. Kjærulff and L.C. van der Gaag. 2000. Making sensitivity analysis computationally efficient. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pages 317 – 325.

K.B. Laskey. 1995. Sensitivity analysis for probability assessments in Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 25: 901 – 909.

P. Liu, L. Lei and N. Wu. 2005. A quantitative study of the effect of missing data in classifiers. In *IEEE Proceedings of the 5th International Conference on Computer and Information Technology*, pages 28 – 33.

I. Rish. 2001. An empirical study of the naive Bayes classifier. In *IJCAI Workshop on Empirical Methods in Artificial Intelligence*, pages 41 – 46.

L.C. van der Gaag and S. Renooij. 2001. Analysing sensitivity data from probabilistic networks. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pages 530 – 537.

L.C. van der Gaag and S. Renooij. 2006. On the sensitivity of probabilistic networks to reliability characteristics. *Modern Information Processing: From Theory to Applications*, Elsevier, pages 385 – 405.

# Abstraction and Refinement for Solving Continuous Markov Decision Processes

Alberto Reyes[1] and Pablo Ibargüengoytia
Inst. de Inv. Eléctricas
Av. Reforma 113, Palmira, Cuernavaca, Mor., México
{areyes,pibar}@iie.org,mx

L. Enrique Sucar and Eduardo Morales
INAOE
Luis Enrique Erro 1, Sta. Ma. Tonantzintla, Pue., México
{esucar,emorales}@inaoep.mx

**Abstract**

We propose a novel approach for solving continuous and hybrid Markov Decision Processes (MDPs) based on two phases. In the first phase, an initial approximate solution is obtained by partitioning the state space based on the reward function, and solving the resulting discrete MDP. In the second phase, the initial abstraction is refined and improved. States with high variance in their value with respect to neighboring states are partitioned, and the MDP is solved locally to improve the policy. In our approach, the reward function and transition model are learned from a random exploration of the environment, and can work with both, pure continuous spaces; or hybrid, with continuous and discrete variables. We demonstrate empirically the method in several simulated robot navigation problems, with different sizes and complexities. Our results show an approximate optimal solution with an important reduction in state size and solution time compared to a fine discretization of the space.

## 1 Introduction

Markov Decision Processes (MDPs) have developed as a standard method for decision-theoretic planning. Traditional MDP solution techniques have the drawback that they require an explicit state representation, limiting their applicability to real-world problems. Factored representations (Boutilier et al., 1999) address this drawback via compactly specifying the state-space in factored form by using dynamic Bayesian networks or decision diagrams. Such Factored MDPs can be used to represent in a more compact way exponentially large state spaces. The algorithms for planning using MDPs, however, still run in time polynomial in the size of the state space or exponential in the number of state-variables; and do not apply to continuous domains.

Many stochastic processes are more naturally defined using continuous state variables which are solved as Continuous Markov Decision Processes (CMDPs) or general state-space MDPs by analogy with general state-space Markov chains. In this approach the optimal value function satisfies the Bellman fixed point equation:

$$V(s) = max_a[R(s,a) + \gamma \int_{s'} p(s'|s,a)V(s')ds'].$$

Where $s$ represents the state, $a$ a finite set of actions, $R$ the immediate reward, $V$ the expected value, $P$ the transition function, and $\gamma$ a discount factor.

The problem with CMDPs is that if the continuous space is discretized to find a solution, the discretization causes yet another level of ex-

[1] Ph.D. Student at ITESM Campus Cuernavaca, Av. Reforma 182-A, Lomas, Cuernavaca, Mor., México

ponential blow up. This "curse of dimensionality" has limited the use of the MDP framework, and overcoming it has become a relevant topic of research. Existing solutions attempt to replace the value function or the optimal policy with a finite approximation. Two recent methods to solve a CMDPs are known as grid-based MDP discretizations and parametric approximations. The idea behind the grid-based MDPs is to discretize the state-space in a set of grid points and approximate value functions over such points. Unfortunately, classic grid algorithms scale up exponentially with the number of state variables (Bonet and Pearl, 2002). An alternative way is to approximate the optimal value function $V(x)$ with an appropriate parametric function model (Bertsekas and Tsitsiklis, 1996). The parameters of the model are fitted iteratively by applying one step Bellman backups to a finite set of state points arranged on a fixed grid or obtained through Monte Carlo sampling. Least squares criterion is used to fit the parameters of the model. In addition to parallel updates and optimizations, on-line update schemes based on gradient descent, e.g., (Bertsekas and Tsitsiklis, 1996) can be used to optimize the parameters. The disadvantages of these methods are their instability and possible divergence (Bertsekas, 1995).

Several authors, e.g., (Dean and Givan, 1997), use the notions of abstraction and aggregation to group states that are similar with respect to certain problem characteristics to further reduce the complexity of the representation or the solution. (Feng et al., 2004) proposes a state aggregation approach for exploiting structure in MDPs with continuous variables where the state space is dynamically partitioned into regions where the value function is the same throughout each region. The technique comes from POMDPs to represent and reason about linear surfaces effectively. (Hauskrecht and Kveton, 2003) show that approximate linear programming is able to solve not only MDPs with finite states, but also be successfully applied to factored continuous MDPs. Similarly, (Guestrin et al., 2004) presents a framework that also exploits structure to model and solve factored MDPs although he extends the technique to be applied to both discrete and continuous problems in a collaborative setting.

Our approach is related to these works, however it differs on several aspects. First, it is based on *qualitative models* (Kuipers, 1986), which are particularly useful for domains with continuous state variables. It also differs in the way the abstraction is built. We use training data to learn a decision tree for the reward function, from which we deduce an abstraction called *qualitative states*. This initial abstraction is refined and improved via a local iterative process. States with high variance in their value with respect to neighboring states are partitioned, and the MDP is solved locally to improve the policy. At each stage in the refinement process, only one state is partitioned, and the process finishes when any potential partition does not change the policy. In our approach, the reward function and transition model are learned from a random exploration of the environment, and can work with both, pure continuous spaces; or hybrid, with continuous and discrete variables.

We have tested our method in simulated robot navigation problems, in which the state space is continuous, with several scenarios with different sizes and complexities. We compare the solution of the models obtained with the initial abstraction and the final refinement, with the solution of a discrete MDP obtained with a fine discretization of the state space, in terms of differences in policy, value and complexity. Our results show an approximate optimal solution with an important reduction in state size and solution time compared to a fine discretization of the space.

The rest of the paper is organized as follows. The next section gives a brief introduction to MDPs. Section 3 develops the abstraction process and Section 4 the refinement stage. In Section 5 the empirical evaluation is described. We conclude with a summary and directions for future work.

## 2 Markov Decision Processes

A Markov Decision Process (MDP) (Puterman, 1994) models a sequential decision problem, in which a system evolves in time and is controlled by an agent. The system dynamics is governed by a probabilistic transition function that maps states and actions to states. At each time, an agent receives a reward that depends on the current state and the applied action. Thus, the main problem is to find a control strategy or *policy* that maximizes the expected reward over time.

Formally, an MDP is a tuple $M =< S, A, \Phi, R >$, where $S$ is a finite set of states $\{s_1, \ldots, s_n\}$. $A$ is a finite set of actions for all states. $\Phi : A \times S \times S$ is the state transition function specified as a probability distribution. The probability of reaching state $s'$ by performing action $a$ in state $s$ is written as $\Phi(a, s, s')$. $R : S \times A \rightarrow \Re$ is the reward function. $R(s, a)$ is the reward that the agent receives if it takes action $a$ in state $s$. A policy for an MDP is a mapping $\pi : S \rightarrow A$ that selects an action for each state. A solution to an MDP is a policy that maximizes its expected value. For the discounted infinite–horizon case with any given discount factor $\gamma \in [0, 1)$, there is a policy $V^*$ that is optimal regardless of the starting state that satisfies the *Bellman* equation (Bellman, 1957):

$$V^*(s) = max_a\{R(s, a)+\gamma\Sigma_{s' \in S}\Phi(a, s, s')V^*(s')\}$$

Two popular methods for solving this equation and finding an optimal policy for an MDP are: (a) value iteration and (b) policy iteration (Puterman, 1994).

### 2.1 Factored MDPs

A common problem with the MDP formalism is that the state space grows exponentially with the number of domain variables, and its inference methods grow in the number of actions. Thus, in large problems, MDPs becomes impractical and inefficient. Factored representations avoid enumerating the problem state space, producing a more concise representation that makes solving more complex problems tractable.

In a factored MDP, the set of states is described via a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, where each $X_i$ takes on values in some finite domain $Dom(X_i)$. A state $\mathbf{x}$ defines a value $x_i \in Dom(X_i)$ for each variable $X_i$. Thus, the set of states $S = Dom(X_i)$ is exponentially large, making it impractical to represent the transition model explicitly as matrices. Fortunately, the framework of dynamic Bayesian networks (DBN) (Dean and Kanazawa, 1989) gives us the tools to describe the transition model function concisely. In these representations, the post-action nodes (at the time $t + 1$) contain matrices with the probabilities of their values given their parents' values under the effects of an action.

## 3 Qualitative MDPs

### 3.1 Qualitative states

We define a qualitative state space $Q$ as a set of states $q_1, q_2, ..q_n$ that have different utility properties. These properties map the state space into a set of *partitions*, such that each partition corresponds to a group of continuous states with a similar reward value. In a qualitative MDP, a state partition $q_i$ is a region bounded by a set of constraints over the continuous dimensions in the state space. The relational operators used in this approach are $<$ and $\geq$. For example, assuming that the immediate reward is a function of the linear position in a robot navigation domain, a qualitative state could be a region in an $x_0 - x_1$ coordinates system bounded by the constraints: $x_0 \geq val(x_0)$ and $x_1 \geq val(x_1)$, expressing that the current $x_0$ coordinate is limited by the interval $[val(x_0), \infty]$, and the $x_1$ coordinate by the interval $[val(x_1), \infty]$. It is evident that a qualitative state can cover a large number of states (if we consider a fine discretization) with similar properties.

Similarly to the reward function in a factored MDP, the state space $Q$ is represented by a decision tree (Q–tree). In our approach, the decision tree is automatically induced from data. Each leaf in the induced decision tree is labeled with a new qualitative state. Even for leaves with the

same reward value, we assign a different qualitative state value. This generates more states but at the same time creates more guidance that helps produce more adequate policies. States with similar reward are partitioned so each q–state is a continuous region. Figure 1 shows this tree transformation in a two dimensional domain.

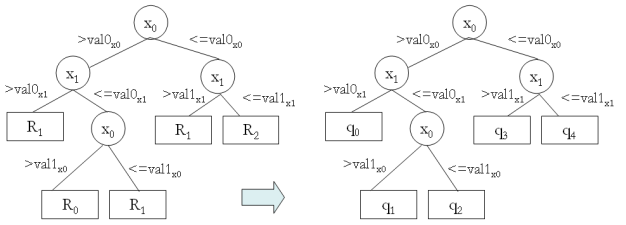


Figure 1: Transformation of the reward decision tree into a Q-tree. Nodes in the tree represent continuous variables and edges evaluate whether this variable is less or greater than a particular bound.

Each branch in the Q–tree denotes a set of constraints for each partition $q_i$. Figure 2 illustrates the constraints associated to the example presented above, and its representation in a 2-dimensional space.

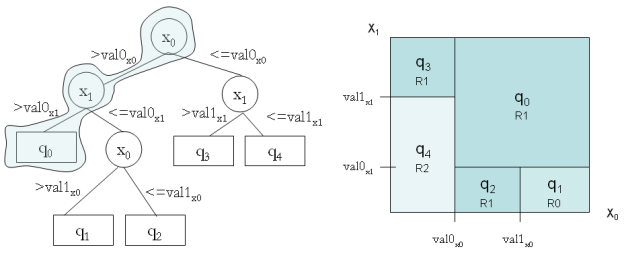


Figure 2: In a Q-tree, branches are constraints and leaves are qualitative states. A graphical representation of the tree is also shown. Note that when an upper or lower variable bound is infinite, it must be understood as the upper or lower variable bound in the domain.

### 3.2 Qualitative MDP Model Specification

We can define a qualitative MDP as a factored MDP with a set of hybrid qualitative–discrete factors. The qualitative state space $Q$, is an additional factor that concentrates all the continuous variables. The idea is to substitute all these variables by this abstraction to reduce the dimensionality of the state space. Initially, only the continuous variables involved in the reward function are considered, but, as described in Section 4, other continuous variables can be incorporated in the refinement stage. Thus, a Qualitative MDP state is described in a factored form as $\mathbf{X} = \{X_1, \ldots, X_n, Q\}$, where $X_1, \ldots, X_n$ are the discrete factors, and $Q$ is a factor that represents the relevant continuous dimensions.

### 3.3 Learning Qualitative MDPs

The Qualitative MDP model is learned from data based on a random exploration of the environment with a 10% Gaussian noise introduced on the actions outcomes. We assume that the agent can explore the state space, and for each state–action can receive some immediate reward. Based on this random exploration, an initial partition, $Q_0$, of the continuous dimensions is obtained, and the reward function and transition functions are induced.

Given a set of state transition represented as a set of random variables, $O^j = \{\mathbf{X_t}, \mathbf{A}, \mathbf{X_{t+1}}\}$, for $j = 1, 2, ..., M$, for each state and action $A$ executed by an agent, and a reward (or cost) $R^j$ associated to each transition, we learn a qualitative factored MDP model:

1. From a set of examples $\{O, R\}$ obtain a decision tree, $RDT$, that predicts the reward function $R$ in terms of continuous and discrete state variables, $X_1, \ldots, X_k, Q$. We used J48, a Java re- implementation of C4.5 (Quinlan, 1993) in Weka, to induce a pruned decision tree.

2. Obtain from the decision tree, $RDT$, the set of constraints for the continuous variables relevant to determine the qualitative states (q–states) in the form of a Q-tree. In terms of the domain variables, we obtain a new variable $Q$ representing the reward-based qualitative state space whose values are the q–states.

3. Qualify data from the original sample in such a way that the new set of attributes are the Q variables, the remaining discrete state variables not included in the decision tree, and the action $A$. This transformed data set can be called the qualified data set.

4. Format the qualified data set in such a way that the attributes follow a temporal causal ordering. For example variable $Q_t$ must be set before $Q_{t+1}$, $X1_t$ before $X1_{t+1}$, and so on. The whole set of attributes should be the variable Q in time t, the remaining system variables in time t, the variable Q in time t+1, the remaining system variables in time t+1, and the action $A$.

5. Prepare data for the induction of a 2-stage dynamic Bayesian net. According to the action space dimension, split the qualified data set into $|A|$ sets of samples for each action.

6. Induce the transition model for each action, $A_j$, using the K2 algorithm (Cooper and Herskovits, 1992).

This initial model represents a high-level abstraction of the continuous state space and can be solved using a standard solution technique, such as value iteration, to obtain the optimal policy. This approach has been successfully applied in some domains. However, in some cases, our abstraction can miss some relevant details of the domain and consequently produce suboptimal policies. We improve this initial partition through a refinement stage described in the next section.

## 4 Qualitative State Refinement

We have designed a value-based algorithm that recursively selects and partitions abtract states with high utility variance. Given an initial partition and a solution for the qualitative MDP, the algorithm proceeds as follows:

While there is an unmarked partition greater than the minimum size:

1. Select an unmarked partition (state) with the highest variance in its utility value with respect to its neighbours

2. Select the dimension (state variable) with the highest difference in utility value with its contiguous states

3. Bisect the dimension (divide the state in two equal-size parts)

4. Solve the new MDP

5. If the new MDP has the same policy as before, mark the original state before the partition and return to the previous MDP, otherwise, accept the refinement and continue.

The minimum size of a state is defined by the user and is domain dependent. For example, in the robot navigation domain, the minimum size depends on the smallest goal (area with certain reward) or on the robot step size. A graphical representation of this process is shown in figure 3. Figure 3 (a) shows an initial partition for two qualitative variables where each qualitative state is a set of ground states with similar reward. Figure 3 (b) shows the refined two-dimension state space after applying the splitting state algorithm.

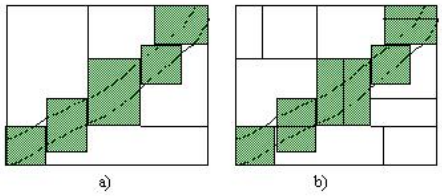



Figure 3: Qualitative refinement for a two-dimension state space. a) initial partition by reward. b) refined state space

## 5 Experimental Results

We tested our approach in a robot navigation domain using a simulated environment. In this setting goals are represented as light-colored square regions with positive immediate reward, while non-desirable regions are represented by

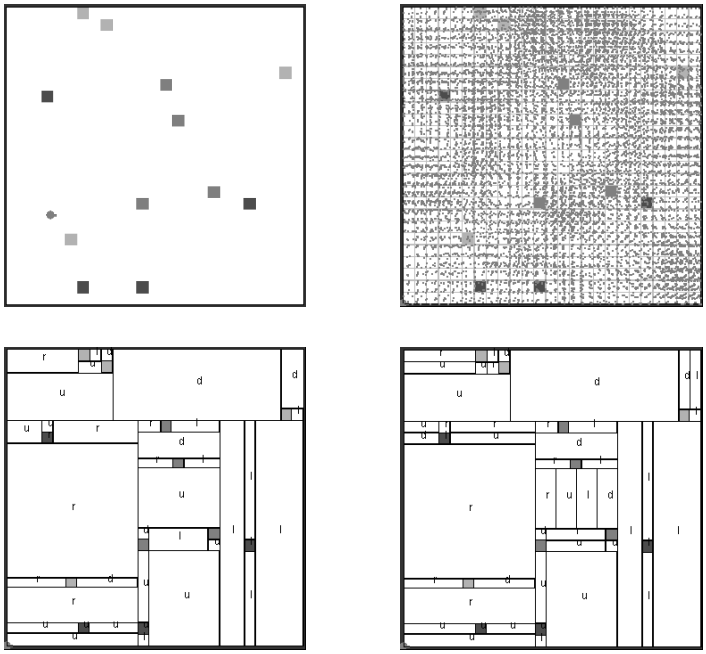


Figure 4: Abstraction and refinement process. Upper left: reward regions. Upper right: exploration process. Lower left: initial qualitative states and their corresponding policies, where u=up, d=down, r=right, and l=left. Lower right: refined partition.

dark-colored squares with negative reward. The remaining regions in the navigation area receive 0 reward (white). Experimentally, we express the size of a rewarded (non zero reward) as a function of the navigation area. Rewarded regions are multivalued and can be distributed randomly over the navigation area. The number of rewarded squares is also variable. Since obstacles are not considered robot states, they are not included.

The robot sensor system included the x-y position, angular orientation, and navigation bounds detection. In a set of experiments the possible actions are discrete orthogonal movements to the right, left, up, and down. Figure 4 upper left shows an example of a navigation problem with 26 rewarded regions. The reward function can have six possible values. In this example, goals are represented as different-scale light colors. Similarly, negative rewards are represented with different-scale dark colors. The planning problem is to automatically obtain an optimal policy for the robot to achieve its goals avoiding negative rewarded regions.

The qualitative representation and refinement were tested with several problems of different sizes and complexities, and compared to a fine discretization of the environment in terms of precision and complexity. The precision is evaluated by comparing the policies and values per state. The *policy precision* is obtained by comparing the policies generated with respect to the policy obtained from a fine discretization. That is, we count the number of *fine* cells in which the policies are the same:

$$PP = (NEC/NTC) \times 100,$$

where $PP$ is the policy precision in percentage, $NEC$ is the number of fine cells with the same policy, and $NTC$ is the total number of fine cells. This measure is pessimistic because in some states it is possible for more than one action to have the same or similar value, and in this measure only one is considered correct.

The utility error is calculated as follows. The utility values of all the states in each representation is first normalized. The sum of the absolute differences of the utility values of the corre-

Table 1: Description of problems and comparison between a "normal" discretization and our qualitative discretization.

| Problem | | | | | Discrete | | | | Qualitative | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Learning | | Inference | | Learning | | Inference | |
| id | no. reward cells | reward size (% dim) | no. reward values | no. samples | no. states | time (ms) | no. iterations | time (ms) | no. states | time (ms) | no. iterations | time (ms) |
| 1 | 2 | 20 | 3 | 40,000 | 25 | 7,671 | 120 | 20 | 8 | 2,634 | 120 | 20 |
| 2 | 4 | 20 | 5 | 40,000 | 25 | 1,763 | 123 | 20 | 13 | 2,423 | 122 | 20 |
| 3 | 10 | 10 | 3 | 40,000 | 100 | 4,026 | 120 | 80 | 26 | 2,503 | 120 | 20 |
| 4 | 6 | 5 | 3 | 40,000 | 400 | 5,418 | 120 | 1,602 | 24 | 4,527 | 120 | 40 |
| 5 | 10 | 5 | 5 | 28,868 | 400 | 3,595 | 128 | 2,774 | 29 | 2,203 | 127 | 60 |
| 6 | 12 | 5 | 4 | 29,250 | 400 | 7,351 | 124 | 7,921 | 46 | 2,163 | 124 | 30 |
| 7 | 14 | 3.3 | 9 | 50,000 | 900 | 9,223 | 117 | 16,784 | 60 | 4,296 | 117 | 241 |

sponding states is evaluated and averaged over all the differences.

Figure 4 shows the abstraction and refinement process for the motion planning problem presented above. A color inversion and gray scale format is used for clarification. The upper left figure shows the rewarded regions. The upper right figure illustrates the exploration process. The lower left figure shows the initial qualitative states and their corresponding policies. The lower right figure shows the refined partition.

Table 1 presents a comparison between the behavior of seven problems solved with a simple discretization approach and our qualitative approach. Problems are identified with a number as shown in the first column. The first five columns describe the characteristics of each problem. For example, problem 1 (first row) has 2 reward cells with values different from zero that occupy 20% of the number of cells, the different number of reward values is 3 (e.g., -10, 0 and 10) and we generated 40,000 samples to build the MDP model.

Table 2 presents a comparison between the qualitative and the refined representation. The first three columns describe the characteristics of the qualitative model and the following describe the characteristics with our refinement process. They are compared in terms of utility error in %, the policy precision also in %, and the time spent in the refinement in minutes.

As can be seen from Table 1, there is a significant reduction in the complexity of the problems using our abstraction approach. This can be clearly appreciated from the number of states and processing time required to solve the problems. This is important since in complex domains where it can be difficult to define an adequate abstraction or solve the resulting MDP problem, one option is to create abstractions and hope for suboptimal policies. To evaluate the quality of the results Table 2 shows that the proposed abstraction produces on average only 9.17% error in the utility value when compared against the values obtained from the dicretized problem as can be seen in Table 2. Finally, since an initial refinement can miss some relevant aspects of the domain, a refinement process may be necessary to obtain more adequate policies. Our refinement process is able to maintain or improve the utility values in all cases. The percentage of policy precision with respect to the initial qualitative model can sometimes decrease with the refinement process. This is due to our pessimistic measure.

Table 2: Comparative results between the initial abstraction and the proposed refinement process.

| | Qualitative | | Refinement | | |
|---|---|---|---|---|---|
| id | util. error (%) | policy precis. (%) | util. error (%) | policy precis. (%) | refin. time (min) |
| 1 | 7.38 | 80 | 7.38 | 80 | 0.33 |
| 2 | 9.03 | 64 | 9.03 | 64 | 0.247 |
| 3 | 10.68 | 64 | 9.45 | 74 | 6.3 |
| 4 | 12.65 | 52 | 8.82 | 54.5 | 6.13 |
| 5 | 7.13 | 35 | 5.79 | 36 | 1.23 |
| 6 | 11.56 | 47.2 | 11.32 | 46.72 | 10.31 |
| 7 | 5.78 | 44.78 | 5.45 | 43.89 | 23.34 |

## 6 Conclusions and Future Work

In this paper, a new approach for solving continuous and hybrid infinite-horizon MDPs is described. In the first phase we use an exploration strategy of the environment and a machine learning approach to induce an initial state abstraction. We then follow a refinement process to improve on the utility value. Our approach creates significant reductions in space and time allowing to solve quickly relatively large problems. The utility values on our abstracted representation are reasonably close (less than 10%) to those obtained using a fine discretization of the domain. A new refinement process to improve the results of the initial proposed abstraction is also presented. It always improves or at least maintains the utility values obtained from the qualitative abstraction. Although tested on small solvable problems for comparison purposes, the approach can be applied to more complex domains where a simple dicretization approach is not feasible.

As future research work we will like to improve our refinement strategy to select a better segmentation of the abstract states and use an alternative search strategy. We also plan to test our approach in other domains.

### Acknowledgments

This work was supported in part by the *Instituto de Investigaciones Eléctricas*, México and CONACYT Project No. 47968.

## References

R.E. Bellman. 1957. *Dynamic Programming.* Princeton U. Press, Princeton, N.J.

D. P. Bertsekas and J.N. Tsitsiklis. 1996. *Neuro-dynamic programming.* Athena Sciences.

D. P. Bertsekas. 1995. A counter-example to temporal difference learning. *Neural Computation*, 7:270–279.

B. Bonet and J. Pearl. 2002. Qualitative mdps and pomdps: An order-of-magnitude approach. In *Proceedings of the 18th Conf. on Uncertainty in AI, UAI-02*, pages 61–68, Edmonton, Canada.

C. Boutilier, T. Dean, and S. Hanks. 1999. Decision-theoretic planning: structural assumptions and computational leverage. *Journal of AI Research*, 11:1–94.

G. F. Cooper and E. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning.*

T. Dean and R. Givan. 1997. Model minimization in Markov Decision Processes. In *Proc. of the 14th National Conf. on AI*, pages 106–111. AAAI.

T. Dean and K. Kanazawa. 1989. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150.

Z. Feng, R. Dearden, N. Meuleau, and R. Washington. 2004. Dynamic programming for structured continuous Markov decision problems. In *Proc. of the 20th Conf. on Uncertainty in AI (UAI-2004). Banff, Canada.*

C. Guestrin, M. Hauskrecht, and B. Kveton. 2004. Solving factored mdps with continuous and discrete variables. In *Twentieth Conference on Uncertainty in Artificial Intelligence (UAI 2004)*, Banff, Canada.

M. Hauskrecht and B. Kveton. 2003. Linear program approximation for factored continuous-state Markov decision processes. In *Advances in Neural Information Processing Systems NIPS(03)*, pages 895– 902.

B. Kuipers. 1986. Qualitative simulation. *AI*, 29:289–338.

M.L. Puterman. 1994. *Markov Decision Processes.* Wiley, New York.

J.R. Quinlan. 1993. *C4.5: Programs for machine learning.* Morgan Kaufmann, San Francisco, Calif., USA.

# Bayesian Model Averaging of TAN Models for Clustering

Guzmán Santafé, Jose A. Lozano and Pedro Larrañaga
Intelligent Systems Group, Computer Science and A. I. Dept.
University of the Basque Country, Spain

**Abstract**

Selecting a single model for clustering ignores the uncertainty left by finite data as to which is the correct model to describe the dataset. In fact, the fewer samples the dataset has, the higher the uncertainty is in model selection. In these cases, a Bayesian approach may be beneficial, but unfortunately this approach is usually computationally intractable and only approximations are feasible. For supervised classification problems, it has been demonstrated that model averaging calculations, under some restrictions, are feasible and efficient. In this paper, we extend the expectation model averaging (EMA) algorithm originally proposed in Santafé et al. (2006) to deal with model averaging of naive Bayes models for clustering. Thus, the extended algorithm, EMA-TAN, allows to perform an efficient approximation for a model averaging over the class of tree augmented naive Bayes (TAN) models for clustering. We also present some empirical results that show how the EMA algorithm based on TAN outperforms other clustering methods.

## 1 Introduction

Unsupervised classification or clustering is the process of grouping similar objects or data samples together into natural groups called clusters. This process generates a partition of the objects to be classified. Bayesian networks (Jensen, 2001) are powerful probabilistic graphical models that can be used for clustering purposes. The naive Bayes is the most simple Bayesian network model (Duda and Hart, 1973). It assumes that the predictive variables in the model are independent given the value of the cluster variable. Despite this being a very simple model, it has been successfully used in clustering problems (Cheeseman and Stutz, 1996). Other models have been proposed in the literature in order to relax the heavy independence assumptions that the naive Bayes model makes. For example, tree augmented naive Bayes (TAN) models (Friedman et al., 1997) allow the predictive variables to form a tree and the class variable remains as a parent of each predictive variable. On the other hand, more complicated methods have been proposed to allow the encoding of context-specific (in)dependencies in clustering problems by using, for instance, naive Bayes (Barash and Friedman, 2002) or recursive Bayesian multinets (Peña et al., 2002) which are trees that incorporate Bayesian multinets in the leaves.

The process of selecting a single model for clustering ignores model uncertainty and it can lead to the selection of a model that does not properly describe the data. In fact, the fewer samples the dataset has, the higher the uncertainty is in model selection. In these cases, a Bayesian approach may be beneficial. The Bayesian approach proposes an averaging over all models weighted by their posterior probability given the data (Madigan and Raftery, 1994; Hoeting et al., 1999). The Bayesian model averaging has been seen by some authors as a model combination technique (Domingos, 2000; Clarke, 2003) and it does not always perform as successfully as expected. However, other authors states that Bayesian model averaging is not exactly a model ensemble technique but a method for 'soft model selection' (Minka, 2002).

Although model averaging approach is normally preferred when there are a few data samples because it deals with uncertainty in model selection, this approach is usually intractable and only approximations are feasible. Typically, the Bayesian model averaging for clustering is approximated by averaging over some of

the models with the highest posterior probabilities (Friedman, 1998). However, efficient calculation of model averaging for supervised classification models under some constraints has been proposed in the literature (Dash and Cooper, 2004; Cerquides and López de Mántaras, 2005). Some of these proposals have also been extended to clustering problems. For example, Santafé et al. (2006) extend the calculations of naive Bayes models to approximate a Bayesian model averaging for clustering. In that paper, the authors introduce the expectation model averaging (EMA) algorithm, which is a variant of the well-known EM algorithm (Dempster et al., 1977) and allows to deal with the latent cluster variable and then approximate a Bayesian model averaging of naive Bayes models for clustering.

In this paper we use the structural features estimation proposed by Friedman and Koller (2003) and the model averaging calculations for supervised classifiers presented in Dash and Cooper (2004) in order to extend the EMA algorithm to TAN models (EMA-TAN algorithm). In other words, we propose a method to obtain a single Bayesian network model for clustering which approximates a Bayesian model averaging over all possible TAN models. This is possible by setting an ancestral order among the predictive variables. The result of the Bayesian model averaging over TAN models is a single Bayesian network model for clustering (Dash and Cooper, 2004).

The rest of the paper is organized as follows. Section 2 introduces the notation that is used throughout the paper as well as the assumptions that we make. Section 3 describes the EMA algorithm to approximate model averaging over the class of TAN models. Section 4 shows some experimental results with synthetic data that illustrate the behavior of the EMA algorithm. Finally, section 5 presents the conclusions of the paper and future work.

## 2 Notation and Assumptions

In an unsupervised learning problem there is a set of predictive variables, $X_1, \ldots, X_n$, and the latent cluster variable, $C$. The dataset $D = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$ contains data samples $\boldsymbol{x}^{(l)} = \{x_1^{(l)}, \ldots, x_n^{(l)}\}$, with $l = 1, \ldots, N$.

We define a Bayesian network model as $\mathcal{B} = \langle S, \boldsymbol{\theta} \rangle$, where $S$ describes the structure of the model and $\boldsymbol{\theta}$ its parameter set. Using the classical notation in Bayesian networks, $\theta_{ijk}$ (with $k = 1, \ldots, r_i$ and $r_i$ being the number of states for variable $X_i$) represents the conditional probability of variable $X_i$ taking its $k$-th value given that its parents, $\boldsymbol{Pa}_i$, takes its $j$-th configuration. The conditional probability mass function for $X_i$ given the $j$-th configuration of its parents is designated as $\boldsymbol{\theta}_{ij}$, with $j = 1, \ldots, q_i$, where $q_i$ is the number of different states of $\boldsymbol{Pa}_i$. Finally, $\boldsymbol{\theta}_i = (\boldsymbol{\theta}_{i1}, \ldots, \boldsymbol{\theta}_{iq_i})$ denotes the set of parameters for variable $X_i$, and therefore, $\boldsymbol{\theta} = (\boldsymbol{\theta}_C, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$, where $\boldsymbol{\theta}_C = (\theta_{C-1}, \ldots, \theta_{C-r_C})$ is the set of parameters for the cluster variable, with $r_C$ the number of clusters fixed in advance.

In order to use the decomposition proposed in Friedman and Koller (2003) for an efficient model averaging calculation, we need to consider an ancestral order $\boldsymbol{\pi}$ over the predictive variables.

**Definition 1.** Class of TAN models ($\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$): given an ancestral order $\boldsymbol{\pi}$, a model $\mathcal{B}$ belongs to $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ if each predictive variable has up to two parents (another predictive variable and the cluster variable) and the arcs between variables that are defined in the structure $S$ are directed down levels: $X_j \rightarrow X_i \in S \Rightarrow level_{\boldsymbol{\pi}}(X_i) < level_{\boldsymbol{\pi}}(X_j)$.

Note that, the classical conception of TAN model allows the predictive variables to form a tree and then, the cluster variable is set as a parent of each predictive variable. However, we do not restrict $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ to only these tree models, but we also allow the predictive variables to form a forest and also the class variable may or may not be set as a parent of each predictive variable. Therefore $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ also includes, among others, naive Bayes and selective naive Bayes models.

For a given ordering $\boldsymbol{\pi}$ and a particular variable $X_i$, we can enumerate all the possible parent sets for $X_i$ in the class of TAN models $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$. In order to clarify calculations, we superscript with $v$ any quantity related to a predictive variable $X_i$ and thus, we are able to identify the parent set of variable $X_i$ that

we are taking into consideration. For example, for a given order $\boldsymbol{\pi} = \langle X_1, X_2, X_3\rangle$ the possible sets of parents for $X_3$ in $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$ are: $\boldsymbol{Pa}^1_3 = \{\emptyset\}, \boldsymbol{Pa}^2_3 = \{X_1\}, \boldsymbol{Pa}^3_3 = \{X_2\}, \boldsymbol{Pa}^4_3 = \{C, X_1\}, \boldsymbol{Pa}^5_3 = \{C, X_2\}, \boldsymbol{Pa}^6_3 = \{C\}$ with, in this case, $v = 1, \ldots, 6$. In general, we consider, without loss of generality, $\boldsymbol{\pi} = \langle X_1, \ldots, X_n\rangle$ and therefore, for a variable $X_i$, $v = 1, \ldots, 2i$. Moreover, we use $i$ to index any quantity related to the $i$-th predictive variable, with $i = 1, \ldots, n$.

Additionally, the following five assumptions are needed to perform an efficient approximation of model averaging over $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$:

**Multinomial variables:** Each variable $X_i$ is discrete and can take $r_i$ states. The cluster variable is also discrete and can take $r_C$ possible states, $r_C$ being the number of clusters fixed in advance.

**Complete dataset:** We assume that there are no missing values for the predictive variables in the dataset. However, the cluster variable is latent; therefore, its values are always missing.

**Dirichlet priors:** The parameters of every model are assumed to follow a Dirichlet distribution. Thus, $\alpha_{ijk}$ is the Dirichlet hyperparameter for parameter $\theta_{ijk}$ from the network, and $\alpha_{C-j}$ is the hyperparameter for $\theta_{C-j}$. In fact, as we have to take into consideration each possible model in $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$, the parameters of the models can be denoted as $\theta^v_{ijk}$. Hence, we assume the existence of hyperparameters $\alpha^v_{ijk}$.

**Parameter independence:** The probability of having the set of parameters $\boldsymbol{\theta}$ for a given structure $S$ can be factorized as follows:

$$p(\boldsymbol{\theta}|S) = p(\boldsymbol{\theta}_C) \prod_{i=1}^{n} \prod_{j=1}^{q_i} p(\boldsymbol{\theta}_{ij}|S) \tag{1}$$

**Structure modularity:** The prior probability $p(S)$ can be decomposed in terms of each variable and its parents:

$$p(S) \propto p_S(C) \prod_{i=1}^{n} p_S(X_i, \boldsymbol{Pa}_i) \tag{2}$$

where $p_S(X_i, \boldsymbol{Pa}_i)$ is the information contributed by variable $X_i$ to $p(S)$, and $p_S(C)$ is the information contributed by the cluster variable.

Parameter independence assumes that the prior on parameters $\theta_{ijk}$ for a variable $X_i$ depends only on local structures. This is known as parameter modularity (Heckerman et al., 1995). Therefore, we can state that for any two network structures $S_1$ and $S_2$, if $X_i$ has the same parent set in both structures then $p(\theta_{ijk}|S_1) = p(\theta_{ijk}|S_2)$. As a consequence, parameter calculations for a variable $X_i$ will be the same in every model whose structure defines that the variable $X_i$ has the same parent set.

## 3 The EMA-TAN Algorithm

The EMA algorithm was originally introduced in Santafé et al. (2006) for dealing with model averaging calculations of naive Bayes models. In this section we present the extension of this algorithm (EMA-TAN) in order to average over TAN models.

**Theorem 1** (Dash and Cooper, 2004)**.** *There exist, for supervised classification problems, a single model $\tilde{\mathcal{B}} = \langle \tilde{S}, \tilde{\boldsymbol{\theta}}\rangle$ which defines a joint probability distribution $p(c, \boldsymbol{x}|\tilde{\mathcal{B}})$ equivalent to the joint probability distribution produced by model averaging over all TAN models. This model $\tilde{\mathcal{B}}$ is a complete Bayesian network where the structure $\tilde{S}$ defines the relationship between variables in such a way that, for a variable $X_i$, the parent set of $X_i$ in the model $\tilde{\mathcal{B}}$ is $\tilde{\boldsymbol{Pa}}_i = \cup_{v=1}^{2i} \boldsymbol{Pa}^v_i$.*

This result can be extended to clustering problems by means of the EMA-TAN algorithm. However, the latent cluster variable prevents the exact calculation of the model averaging and therefore the model $\tilde{\mathcal{B}}$ obtained by the EMA-TAN algorithm is not an exact model averaging over $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$ but an approximation. Therefore, the EMA-TAN algorithm provides a powerful tool that allows to learn a single unsupervised Bayesian network model which approximates Bayesian model averaging over $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$.

The unsupervised classifier is obtained by learning the predictive probability, $p(c, \boldsymbol{x}|D)$, averaged over the *maximum a posteriori* (MAP) parameter configurations for all the models in $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$. Then, we can obtain the unsupervised classifier by using the conditional probability of the cluster variable given by Bayes' rule.

The EMA-TAN algorithm is an adaption of

the well-known EM algorithm. It uses the E step of the EM algorithm to deal with the missing values for the cluster variable. Then, it performs a model averaging step (MA) to obtain $p(c, \boldsymbol{x}|D)$ and thus the unsupervised Bayesian network model for clustering.

The EMA-TAN, as well as the EM algorithm, is an iterative process where the two steps of the algorithm are repeated successively until a stopping criterion is met. At the $t$-th iteration of the algorithm, a set of parameters, $\tilde{\boldsymbol{\theta}}^{(t)}$, for the Bayesian network model $\tilde{\mathcal{B}}^{(t)}$ is calculated. Note that, although we differentiate between the Bayesian network models among the iterations of the EMA-TAN algorithm, the structure of the model, $\tilde{\boldsymbol{S}}$, is constant and only the estimated parameter set changes. The algorithm stops when the difference between the sets of parameters learned in two consecutive iterations, $\tilde{\boldsymbol{\theta}}^{(t)}$ and $\tilde{\boldsymbol{\theta}}^{(t+1)}$, is less than threshold $\epsilon$, which is fixed in advance.

In order to use the EMA-TAN algorithm, we need to set an initial parameter configuration, $\tilde{\boldsymbol{\theta}}^{(0)}$, and the value of $\epsilon$. The values for $\tilde{\boldsymbol{\theta}}^{(0)}$ are usually taken at random and $\epsilon$ is set at a small value. Note that, even though the obtained model is a single unsupervised Bayesian network, its parameters are learned taking into account the MAP parameter configuration for every model in $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$. Thus, the resulting unsupervised Bayesian network will incorporate into its parameters information about the (in)dependencies between variables described by the different TAN models.

### 3.1 E Step (Expectation)

Intuitively, we can see this step as a *completion* of the values for the cluster variable, which are missing. Actually, this step computes the expected sufficient statistics in the dataset for variable $X_i$ in every model in $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$ given the current model, $\tilde{\mathcal{B}}^{(t)}$. These expected sufficient statistics are used in the next step of the algorithm, MA, as if they were actual sufficient statistics from a complete dataset. From now on, $D^{(t)}$ denotes the dataset after the E step at the $t$-th iteration of the algorithm.

Note that, due to parameter modularity, we do not actually need to calculate the expected sufficient statistics for all the models in $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$ because some of these models share the same value for the expected sufficient statistics. Hence, it is only necessary to calculate the expected sufficient statistics with different parent sets. They can be obtained as follows:

$$E(N^v_{ijk}|\tilde{\mathcal{B}}^{(t)}) = \sum_{l=1}^{N} p(x_i^k, \boldsymbol{Pa}_i^v = j|\boldsymbol{x}^{(l)}, \tilde{\mathcal{B}}^{(t)}) \quad (3)$$

where $x_i^k$ represents the $k$-th value of the $i$-th variable. The expected sufficient statistic $E(N^v_{ijk}|\tilde{\mathcal{B}}^{(t)})$ denotes, at iteration $t$, the expected number of cases in the dataset $D$ where variable $X_i$ takes its $k$-th value, and the $v$-th parent set of $X_i$ takes its $j$-th configuration.

Similarly, we can obtain the expected sufficient statistics for the cluster variable. This is a special case since for any model in $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$ the parent set for $C$ is the same (the cluster variable does not have any parent). Therefore, we refuse the use of superindex $v$ in those quantities related only to $C$.

$$E(N_{C-j}|\tilde{\mathcal{B}}^{(t)}) = \sum_{l=1}^{N} p(C = j|\boldsymbol{x}^{(l)}, \tilde{\mathcal{B}}^{(t)}) \quad (4)$$

Note that, some of the expected sufficient statistics $E(N^v_{ijk}|\tilde{\mathcal{B}}^{(t)})$ do not depend on the value of $C$. Therefore, these values are constant throughout the iterations of the algorithm and it is necessary to calculate them only once.

### 3.2 MA Step (Model Averaging)

In this second step, the EMA algorithm performs the model averaging calculations which obtain a single Bayesian network model with parameters $\tilde{\boldsymbol{\theta}}^{(t+1)}$. These parameters are obtained by calculating $p(c, \boldsymbol{x}|D^{(t)})$ as an average over the MAP configurations for the models in $\mathcal{L}^{\boldsymbol{\pi}}_{TAN}$.

In order to make the calculations clearer, we first show how we can obtain $p(c, \boldsymbol{x}|S, D^{(t)})$ for a fixed structure $S$:

$$p(c, \boldsymbol{x}|S, D^{(t)}) = \int p(c, \boldsymbol{x}|S, \boldsymbol{\theta}) p(\boldsymbol{\theta}|S, D^{(t)}) d\boldsymbol{\theta} \quad (5)$$

The exact computation of the integral in Equation 5 is intractable for clustering problems, therefore, an approximation is needed

(Heckerman et al., 1995). However, assuming parameter independence and Dirichlet priors, and given that the expected sufficient statistics calculated in the previous E step can be used as an approximation to the actual sufficient statistics in the complete dataset, we can approximate $p(c, \boldsymbol{x}|S, D^{(t)})$ by the MAP parameter configuration. This is the parameter configuration that maximizes $p(\boldsymbol{\theta}|S, D^{(t)})$ and can be described in terms of the expected sufficient statistics and the Dirichlet hyperparameters (Heckerman et al., 1995; Cooper and Herskovits, 1992). Therefore, Equation 5 results:

$$p(c, \boldsymbol{x}|S, D^{(t)}) \approx \frac{\alpha_{C-j} + E(N_{C-j}|\tilde{\mathcal{B}}^{(t)})}{\alpha_C + E(N_C|\tilde{\mathcal{B}}^{(t)})} \cdot \quad (6)$$

$$\prod_{i=1}^{n} \frac{\alpha_{ijk}^{\mu_i} + E(N_{ijk}^{\mu_i}|\tilde{\mathcal{B}}^{(t)})}{\alpha_{ij}^{\mu_i} + E(N_{ij}^{\mu_i}|\tilde{\mathcal{B}}^{(t)})} = \hat{\theta}_{C-j} \prod_{i=1}^{n} \hat{\theta}_{ijk}^{\mu_i}$$

where $\hat{\theta}_{ijk}^{\mu_i}$ is the MAP parameter configuration for $\theta_{ijk}^{\mu_i}$ ($\mu_i$ denotes the parent index that corresponds to the parent set for $X_i$ described in $S$), $\alpha_{ij}^{\mu_i} = \sum_{k=1}^{r_i} \alpha_{ijk}^{\mu_i}$, $E(N_{ij}|\tilde{\mathcal{B}}^{(t)}) = \sum_{k=1}^{r_i} E(N_{ijk}|\tilde{\mathcal{B}}^{(t)})$ and similarly for the values related to $C$.

Considering that the structure is not fixed *a priori*, we should average over all selective model structures in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ in the following way:

$$p(c, \boldsymbol{x}|D^{(t)}) = \quad (7)$$

$$\sum_S \int p(c, \boldsymbol{x}|S, \boldsymbol{\theta}) p(\boldsymbol{\theta}|S, D^{(t)}) d\boldsymbol{\theta} \quad p(S|D^{(t)})$$

Therefore, the model averaging calculations require a summation over $2^n n!$ terms, which are the models in $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$.

Using the previous calculations for a fixed structure, Equation 8 can be written as:

$$p(c, \boldsymbol{x}|D^{(t)}) \approx \sum_S \hat{\theta}_{C-j} \prod_{i=1}^{n} \hat{\theta}_{ijk}^{\mu_i} \quad p(S|D^{(t)})$$

$$\propto \sum_S \hat{\theta}_{C-j} \prod_{i=1}^{n} \hat{\theta}_{ijk}^{\mu_i} \quad p(D^{(t)}|S) \quad p(S) \quad (8)$$

Given the assumption of Dirichlet priors and parameter independence, we can approximate $p(D^{(t)}|S)$ efficiently. In order to do so, we adapt the formula to calculate the marginal likelihood with complete data (Cooper and Herskovits, 1992) to our problem with missing values. Thus, we have an approximation to $p(D^{(t)}|S)$:

$$p(D^{(t)}|S) \approx \frac{\Gamma(\alpha_C)}{\Gamma(\alpha_C + E(N_C|\tilde{\mathcal{B}}^{(t)}))} \prod_{j=1}^{r_C} \frac{\Gamma(\alpha_{C-j} + E(N_{C-j}|\tilde{\mathcal{B}}^{(t)}))}{\Gamma(\alpha_{C-j})} \cdot$$

$$\prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij}^{\mu_i})}{\Gamma(\alpha_{ij}^{\mu_i} + E(N_{ij}^{\mu_i}|\tilde{\mathcal{B}}^{(t)}))} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}^{\mu_i} + E(N_{ijk}^{\mu_i}|\tilde{\mathcal{B}}^{(t)}))}{\Gamma(\alpha_{ijk}^{\mu_i})}$$

At this point, given structure modularity assumption, we are able to approximate $p(c, \boldsymbol{x}|D^{(t)})$ with the following expression:

$$p(c, \boldsymbol{x}|D^{(t)}) \approx \kappa \sum_S \rho_{C-j} \prod_{i=1}^{n} \rho_{ijk}^{\mu_i} \quad (9)$$

where $\kappa$ is a constant and $\rho_{C-j}$ and $\rho_{ijk}^{\mu_i}$ are defined in Equation 10.

$$\rho_{C-j} = \hat{\theta}_{C-j} \quad p_S(C) \frac{\Gamma(\alpha_C)}{\Gamma(\alpha_C + E(N_C|\mathcal{B}^{(t)}))} \cdot$$

$$\prod_{j=1}^{r_C} \frac{\Gamma(\alpha_{C-j} + E(N_{C-j}|\mathcal{B}^{(t)}))}{\Gamma(\alpha_{C-j})} \quad (10)$$

$$\rho_{ijk}^{\mu_i} = \hat{\theta}_{ijk}^{\mu_i} \quad p_S(X_i, \boldsymbol{Pa}_i^{\mu_i}) \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij}^{\mu_i})}{\Gamma(\alpha_{ij}^{\mu_i} + E(N_{ij}^{\mu_i}|\mathcal{B}^{(t)}))} \cdot$$

$$\prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}^{\mu_i} + E(N_{ijk}^{\mu_i}|\mathcal{B}^{(t)}))}{\Gamma(\alpha_{ijk}^{\mu_i})}$$

Since we are assuming parameter independence, structure modularity and parameter modularity, we can apply the dynamic programming solution described in Friedman and Koller (2003), and Dash and Cooper (2004). Thus Equation 9 can be written as follows:

$$p(c, \boldsymbol{x}|D^{(t)}) \approx \lambda \quad \rho_{C-j} \prod_{i=1}^{n} \sum_{v=1}^{2i} \rho_{ijk}^{v} \quad (11)$$

with $\lambda$ being a constant.

Note that the time complexity needed to calculate the averaging over $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ is the same as that which is needed to learn the MAP parameter configuration for $\tilde{\mathcal{B}}$.

We can see the similarity of the above-described Equation 11 with the factorization of a Bayesian network model. Indeed the joint probability distribution of the approximated Bayesian model averaging over $\mathcal{L}_{TAN}^{\boldsymbol{\pi}}$ for clustering is equivalent to a single Bayesian network model. Therefore, the parameters of the model for the next iteration of the algorithm can be calculated as follows:

$$\tilde{\theta}_{C-j}^{(t+1)} \propto \rho_{C-j} \quad , \quad \tilde{\theta}_{ijk}^{(t+1)} \propto \sum_{v=1}^{2i} \rho_{ijk}^{v} \qquad (12)$$

### 3.3 Multi-start EMA-TAN

The EMA-TAN is a greedy algorithm that is susceptible to be trapped in a local optima. The results obtained by the algorithm depend on the random initialization of the parameters. Therefore, we propose the use of a multi-start scheme where $m$ different runs of the algorithm with different random initializations are performed. In Santaf et al. (2006) different criteria to obtain the final model from the multi-start process are proposed. In our case, we use the same criteria that the multi-start EM uses: the best model in terms of likelihood among all the $m$ calculated models is selected as the final model. This is not a pure Bayesian approach to the model averaging process but, in practice, it works as well as other more complicated techniques.

## 4 Evaluation in Clustering Problems

It is not easy to validate clustering algorithms since clustering problems do not normally provide information about the true grouping of data samples. In general, it is quite common to use synthetic data because the true model that generated the dataset as well as the underlying clustering structure of the data are known. In order to illustrate the behavior of the EMA-TAN algorithm, we compare it with the classical EM algorithm and with the EMA algorithm (Santafé et al., 2006). For EMA-TAN evaluation, we obtain random TAN models where the number of predictive variables vary in $\{2, 4, 8, 10, 12, 14\}$, each predictive variable can take up to three states and the number of clusters is set to two. For each TAN configuration we generate 100 random models and each one of these models is sampled to obtain different datasets of sizes $40, 80, 160, 320$ and 640. In the experiments, we compare the multi-start EMA-TAN (called also EMA-TAN for convenience) with three different algorithms:

**EM-TAN:** a multi-start EM that learns a TAN model by using, at each step of the EM algorithm, the classical method proposed by Friedman et al. (1997) adapted to be used within the EM algorithm.

**EM-BNET:** a multi-start EM algorithm used to learn the MAP parameters of a complete Bayesian network for a given order $\boldsymbol{\pi}$.

**EMA:** the multi-start model averaging of naive Bayes for clustering.

Note that, for a given order $\boldsymbol{\pi}$, both EMA-TAN and EM-BNET models share the same network structure but their parameter sets are calculated in a different way. The number of multi-start iterations for both multi-start EM and multi-start EMA is $m = 100$.

Since the datasets are synthetically generated, we are able to know the real ordering among variables. Nevertheless, we prefer to use a more general approach and assume that this order is unknown. Therefore, we use a random ordering among predictive variables for each EMA-TAN model that we learn. As the EM-BNET algorithm also needs an ancestral order among predictive variables, in the experiments, we use the same random ordering for any pair of models (EMA-TAN vs. EM-BNET) that we compare.

Every model is used to cluster the dataset from which it has been learned. In the experiments, we compare EMA-TAN vs. EM-TAN, EMA-TAN vs. EM-BNET and EMA-TAN vs. EMA. For each test, the winner model is obtained by comparing the data partitions obtained by both models with the true partition of the dataset. Thus, the model with the best estimated accuracy (percentage of correctly classified instances) is the winner model. In Table 1, the results from the experiments with random TAN models are shown. For each model configuration, the table describes the number of wins/draws/losses of the EMA-TAN models with respect to EM-TAN, EM-BNET or EMA models on basis of the estimated accuracy of each model. We also provide information about

a Wilcoxon signed-rank test[1] used to evaluate whether the accuracy estimated by two different models is different at the 1% and 10% levels. We write the results shown in Table 1 in bold if the test is surpassed at the 10% level and inside a gray color box if the test is surpassed at the 1% level. It can be seen that, in general, the EMA-TAN models behave better than the others and the differences between estimated accuracy are, in most of the cases, statistically significant.

We can see that the compared models obtain very similar results when they have a few predictive variables. This is because the set of models that we are averaging over to obtain the EMA-TAN model is very small. Therefore, it is quite possible that other algorithms such as EM-TAN select the correct model. Hence, in some experiments with the simplest models (models with 2 predictive variables), EMA-TAN algorithm significantly lost with the other algorithms. However, when the number of predictive variables in the model increases and the dataset size is relatively big (the smallest datasets are not big enough for a reliable estimation of the parameters) the EMA-TAN considerably outperforms any other model in the test.

The experimental results from this section reinforce the idea that the results of the Bayesian model averaging outperform other methods when the model that generated the data is included in the set of models that we are averaging over. Since we are averaging over a restricted class of models, this situation may not be fulfilled when applying the EMA-TAN to real problems. Due to the lack of space, we do not include in the paper more experimentation in progress, but we are aware that it would be very interesting to check the performance of the EMA algorithm with other synthetic datasets generated by sampling naive Bayes and general Bayesian network models and also with dataset from real problems.

## 5 Conclusions

We have shown that it is possible to obtain a single unsupervised Bayesian network that approximates model averaging over the class of TAN models. Furthermore, this approximation can be performed efficiently. This is possible by using the EMA-TAN algorithm. The EMA is an algorithm originally proposed in Santafé et al. (2006) for averaging over naive Bayes models in clustering problems. In this paper we extend the algorithm to deal with more complicated models such as TAN (EMA-TAN algorithm). We also present an empirical evaluation by comparing the model averaging over TAN models with the model averaging over naive Bayes models and with the EM algorithm to learn a single TAN model and a Bayesian network model. These experiments conclude that, at least, when the model that generated the dataset is included in the set of models that we average over, the averaging over TAN models outperforms the other methods

Probably, one of the limitations of the proposed algorithm is that, because the learned model which approximates model averaging is a complete Bayesian network, it is computationally hard to learn the model for problems with many predictive variables. In order to overcome this situation, we can restrict the final model to a maximum of $k$ possible parents for each predictive variable. Future work might include a more exhaustive empirical evaluation of the EMA-TAN algorithm and the application of the algorithm to real problems. Moreover, the algorithm can be extended to other Bayesian classifiers such as k-dependent naive Bayes (kDB), selective naive Bayes, etc. Another interesting future work may be the relaxation of complete dataset assumption.

[1]Since we have checked by means of a Kolmogorov-Smirnov test that not all the outcomes from the experiment can be considered to follow a normal distribution, we decided to use a Wilcoxon sign-rank test to compare the results


## Acknowledgments

This work was supported by the SAIOTEK-Autoinmune (II) 2006 and Etortek research projects from the Basque Government, by the Spanish Ministerio de Educación y Ciencia under grant TIN 2005-03824, and by the Government of Navarre under a PhD grant.


## References


Y. Barash and N. Friedman. 2002. Context-specific Bayesian clustering for gene expression data. *Journal of Computational Biology*, 9:169–191.

EMA-TAN vs. EM-TAN

| #Var | 40 | 80 | 160 | 320 | 640 |
|---|---|---|---|---|---|
| 2 | 11/71/18 | **9/71/20** | 15/68/17 | 18/70/12 | **15/63/22** |
| 4 | 37/24/39 | **33/17/50** | 50/8/42 | 48/5/47 | 46/3/51 |
| 8 | 51/6/43 | **57/4/39** | **65/6/29** | **74/4/22** | **71/1/28** |
| 10 | 48/5/47 | **59/4/37** | **55/10/35** | **64/6/30** | **67/4/29** |
| 12 | **54/4/42** | **65/3/32** | **65/4/31** | **69/4/27** | **83/2/15** |
| 14 | **59/8/33** | **68/6/26** | **75/2/23** | **76/4/20** | **76/6/18** |

EMA-TAN vs. EM-BNET

| #Var | 40 | 80 | 160 | 320 | 640 |
|---|---|---|---|---|---|
| 2 | 24/51/25 | 29/35/36 | 33/34/33 | 28/35/37 | 41/29/30 |
| 4 | **54/11/35** | 51/8/41 | **59/8/33** | 49/6/45 | **57/1/42** |
| 8 | **59/8/33** | **64/5/31** | **81/1/18** | **83/0/17** | **91/0/9** |
| 10 | **67/7/26** | **76/0/24** | **82/2/16** | **84/0/16** | **94/0/6** |
| 12 | **66/5/29** | **86/1/13** | **80/0/20** | **91/0/9** | **89/0/11** |
| 14 | **74/2/24** | **83/1/16** | **92/1/7** | **92/0/8** | **96/0/4** |

EMA-TAN vs. EMA

| #Var | 40 | 80 | 160 | 320 | 640 |
|---|---|---|---|---|---|
| 2 | 21/53/26 | **24/41/35** | 24/48/28 | 23/45/32 | 25/45/30 |
| 4 | 47/14/39 | 49/6/45 | 49/10/41 | 50/5/45 | 56/3/41 |
| 8 | **52/10/38** | **58/4/38** | **80/4/16** | **90/0/10** | **89/0/11** |
| 10 | 48/13/39 | **55/8/37** | **69/6/25** | **81/1/18** | **88/1/11** |
| 12 | **55/8/37** | **78/6/16** | **79/1/20** | **88/2/10** | **88/0/12** |
| 14 | **55/11/34** | **68/7/25** | **81/3/16** | **84/3/13** | **92/0/8** |

Table 1: Comparison between EMA-TAN models and EM-TAN, EM-BNET and EMA learned from datasets sampled from random TAN models


J. Cerquides and R. López de Mántaras. 2005. TAN classifiers based on decomposable distributions. *Machine Learning*, 59(3):323–354.

P. Cheeseman and J. Stutz. 1996. Bayesian classification (Autoclass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press.

B. Clarke. 2003. Comparing Bayes model averaging and stacking when model approximation error cannot be ignored. *Journal of Machine Learning Research*, 4:683–712.

G. F. Cooper and E. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

D. Dash and G. F. Cooper. 2004. Model averaging for prediction with discrete Bayesian networks. *Journal of Machine Learning Research*, 5:1177–1203.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38.

P. Domingos. 2000. Bayesian averaging of classifiers and the overfitting problem. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 223–230.

R. Duda and P. Hart. 1973. *Pattern Classification and Scene Analysis*. John Wiley and Sons.

N. Friedman and D. Koller. 2003. Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50:95–126.

N. Friedman, D. Geiger and M. Goldszmidt 1997. Bayesian networks classifiers. *Machine Learning*, 29:131–163.

N. Friedman. 1998. The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 129–138.

D. Heckerman, D. Geiger, and D. M. Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243.

J. Hoeting, D. Madigan, A. E. Raftery, and C. Volinsky. 1999. Bayesian model averaging. *Statistical Science*, 14:382–401.

F.V. Jensen. 2001. *Bayesian Networks and Decision Graphs*. Springer Verlag.

D. Madigan and A. E. Raftery. 1994. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89:1535–1546.

T. Minka. 2002. Bayesian model averaging is not model combination. MIT Media Lab note.

J. M. Peña, J. A. Lozano, and P. Larrañaga. 2002. Learning recursive Bayesian multinets for clustering by means of constructive induction. *Machine Learning*, 47(1):63-90.

G. Santafé, J. A. Lozano, and P. Larrañaga. 2006. Bayesian model averaging of naive Bayes for clustering. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*. Accepted for publication.

# Dynamic Weighting A* Search-based MAP Algorithm for Bayesian Networks

Xiaoxun Sun
Computer Science Department
University of Southern California
Los Angeles, CA 90089
xiaoxuns@usc.edu

Marek J. Druzdzel
Decision Systems Laboratory
School of Information Sciences
& Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
marek@sis.pitt.edu

Changhe Yuan
Department of Computer Science
and Engineering
Mississippi State University
Mississippi State, MS 39762
cyuan@cse.msstate.edu

**Abstract**

In this paper we introduce the Dynamic Weighting A* (DWA*) search algorithm for solving MAP. By exploiting asymmetries in the distribution of MAP variables, the algorithm can greatly reduce the search space and yield MAP solutions with high quality.

## 1 Introduction

The Maximum *A-Posteriori* assignment (MAP) is the problem of finding the most probable instantiation of a set of variables given partial evidence on the remaining variables in a Bayesian network. A special case of MAP that has been paid much attention is the Most Probable Explanation (MPE) problem. MPE is the problem of finding the most probable state of the model given that all its evidences have been observed. MAP turns out to be a much harder problem compared to MPE. Particularly, MPE is NP-complete while the corresponding MAP problem is $NP^{PP}$-complete (Park, 2002). MAP is more useful than MPE for providing explanations. For instance, in diagnosis, generally we are only interested in the configuration of fault variables given some observations. There may be many other variables that have not been observed and are outside the scope of our interest.

In this paper, we introduce the Dynamic Weighting A* (DWA*) search algorithm for solving MAP that is generally more efficient than existing algorithms. The algorithm explores the asymmetries among all possible assignments in the joint probability distribution of the MAP variables. Typically, a small fraction of all possible assignments can be expected to cover a large portion of the total probability space with the remaining assignments having practically negligible probability (Druzdzel, 1994). Also, the DWA* uses dynamic weighting based on *greedy guess* (Park and Darwiche, 2001; Yuan et al., 2004) as the heuristic function. While it is theoretically not admissible (admissible heuristic should offer an upper bound on the MAP), it offers $\epsilon$-admissibility and excellent performance in practice (Pearl, 1988).

## 2 MAP

The MAP problem is defined as follows: Let $\mathbf{M}$ be the set of MAP variables (we are interested in the most probable configuration of these variables); $\mathbf{E}$ is the set of evidence, namely the variables whose states we have observed; The remainder of the variables, denoted by $\mathbf{S}$, are variables that are neither observed nor of interest to us. Given an assignment $\mathbf{e}$ of variables $\mathbf{E}$, the MAP problem is that of finding the assignment $\mathbf{m}$ of variables $\mathbf{M}$ which maximizes the probability of $P(m \mid e)$, while MPE is the special case of MAP with $\mathbf{S}$ being empty.

$$map = \max_{M} \sum_{S} p(M, S \mid E) \; . \qquad (1)$$

In Bayesian networks, we use the Conditional Probability Table (CPT) $\phi$ as the *potential* over a variable and its parents. A potential over all

the states of one variable after updating beliefs is called *marginal*. The notation $\phi_e$ stands for the potential in which we have fixed the value of $e \in E$. Then the probability of MAP with $\Phi$ as its CPTs turns out to be a real number:

$$map = \max_{M} \sum_{S} \prod_{\phi \in \Phi} \phi_e \ . \tag{2}$$

In Equation 2, summation does not commute with maximization. Therefore, it is necessary to do summation before the maximization. The order is called the *elimination order*. The size of the largest clique minus 1 in a jointree constructed based on an elimination order is called the *induced width*. The induced width of the best elimination order is called the *treewidth*. However, for the MAP problems in which neither the set **S** nor the set **M** are empty, the order is constrained. Then the constrained elimination order is known as the *constrained treewidth*. Generally, the constrained treewidth is much larger than treewidth, leading the problem beyond the limits of feasibility.

Several researchers have proposed algorithms for solving MAP. A very efficient approximate search-based algorithm based on local search, proposed by Park (2002), is capable of solving MAP efficiently. An exact method, based on branch-and-bound depth-first search, proposed by Park and Darwiche (2003), performs quite well when the search space is not too large. Another approximate scheme, proposed by Yuan et al. (2004), is a Reheated Annealing MAP algorithm. It is somewhat slower on simple networks but it is capable of handling difficult cases that exact methods cannot tackle.

## 3 Solving MAP using Dynamic Weighting A* Search

We propose in this section an algorithm for solving MAP using Dynamic Weighting A* search, which incorporates the *dynamic weighting* (Pearl, 1988) in the heuristic function, *relevance reasoning* (Druzdzel and Suermondt, 1994), and *dynamic ordering* in the search tree.

### 3.1 A* search

MAP can be solved by A* search in the probability tree that is composed of all the variables in the MAP set. The nodes in the search tree represent partial assignments of the MAP variables **M**. The root node represents an empty assignment. Each MAP variable will be instantiated in a certain order. If a variable **x** in the set of MAP variables **M** is instantiated at the i*th* place using its j*th* state, it will be denoted as $\mathbf{M}_{ij}$. Leaves of the search tree correspond to the last MAP variable that has been instantiated. The vector of instantiated states of each MAP variable is called an *assignment* or a *scenario*. We compute the probability of assignments while searching the whole probability tree using chain rule. For each inner node, the newly instantiated node will be added to the evidence set, i.e., the evidence set will be extended to $\mathbf{M}_{ij} \cup \mathbf{E}$. Then the probability of an assignment of **n** MAP variables can be computed as follows:

$$\begin{aligned} P(\mathrm{M} \mid E) \ &= \ P(M_{ni} \mid M_{1j}, M_{2k}, \ldots M_{(n-1)t}, E) \\ &\quad \ldots P(M_{2k} \mid M_{1j}, E) P(M_{1j} \mid E) \ . \end{aligned}$$

Suppose we are in the x*th* layer of the search tree and preparing for instantiating the x*th* MAP variables. Then the function above can be rewritten as follows:

$$\begin{aligned} &P(\mathrm{M} \mid E) = \\ &\overbrace{P(M_{ni} \mid M_{1j} \ldots M_{(n-1)t}, E) \ldots P(M_{(x+1)z} \mid M_{xy} \ldots E)}^{b} \\ &\underbrace{\cdot P(M_{xy} \mid M_{1j}, M_{2k} \ldots M_{(x-1)q}, E) \ldots P(M_{1j} \mid E)}_{a} \end{aligned} \tag{3}$$

The general idea of DWA* is that in each inner node of the probability tree, we can compute the value of item (a) in the function above *exactly*. We can estimate the heuristic value of the item (b) for the MAP variables that have not been instantiated given the initial evidence set and the MAP variables that have been instantiated as the new evidence. In order to fit the typical format of the cost function of A* search, we can take the logarithm of the equation above, which will not change its monotonicity. Then we get $f(n) = g(n) + h(n)$, where $g(n)$ and $h(n)$

are obtained from the logarithmic transformation of items (a) and (b) respectively. $g(n)$ gives the exact cost from the start node to node in the n*th* layer of the search tree, and h(n) is the estimated cost of the best search path from the n*th* layer to the leaf nodes of the search tree. In order to guarantee the optimality of the solution, $h(n)$ should be *admissible*, which in this case means that it should be an upper-bound on the value of any assignment with the currently instantiated MAP variables as its elements.

### 3.2 Heuristic Function with Dynamic Weighting

**Definition 1.** A heuristic function $h_2$ is said to be *more informed than* $h_1$ if both are admissible and $h_2$ is closer to the optimal cost.

For the MAP problem, the probability of the optimal assignment $P_{opt} < h_2 < h_1$.

**Theorem 1.** *If $h_2$ is more informed than $h_1$ then $A_2^*$* dominates *$A_1^*$. (Pearl, 1988)*

The power of the heuristic function is measured by the amount of pruning induced by $h(n)$ and depends on the accuracy of this estimate. If $h(n)$ estimates the completion cost precisely ($h(n) = P_{opt}$), then A* will only expand nodes on the optimal path. On the other hand, if no heuristic at all is used, (for the MAP problem this amounts to $h(n) = 1$), then a uniform-cost search ensues, which is far less efficient. So it is critical for us to find an *admissible* and *tight* $h(n)$ to get both accurate and efficient solutions.

#### 3.2.1 Greedy Guess

If each variable in the MAP set **M** is conditionally independent of all remaining MAP variables (this is called *exhaustive independence*), then the MAP problem amounts to a simple computation based on the *greedy* chain rule. We instantiate the MAP variable in the current search layer to the state with the largest probability and repeat this for each of the remaining MAP variables one by one. The probability of MAP is then

$$P(M|E) = \prod_{i=1}^{n} \max_j P(M_{ij}|M_{(i-1)k} \dots M_{1m}, E) \ . \tag{4}$$

The requirement of exhaustive independence is too strict for most MAP problems. However, simulation results show that in practice, even when this requirement is violated, the product is still extremely close to the MAP probability (Yuan et al., 2004). This suggests using it as an $\epsilon$-admissible heuristic function (Pearl, 1988).

The curve *Greedy Guess Estimate* in Figure 1 shows that with the increase of the number of MAP variables, the ratio between the greedy guess and the accurate estimate of the optimal probability diverges from the ideal ratio *one* although not always monotonically.

#### 3.2.2 Dynamic Weighting

Since greedy guess is a tight lower bound on the optimal probability of MAP, it is possible to compensate for the error between the greedy guess and the optimal probability. We can do this by adding a weight to the greedy guess such that the product of them is equal to or larger than the optimal probability for each inner node in the search tree. This, it turns out, yields an excellent $\epsilon$-*admissible* heuristic function. This assumption can be represented as follows:

$$\exists\epsilon\{\forall P_{GreedyGuess} * (1+\epsilon) \geq P_{opt} \wedge \\ \forall\epsilon^{'}(P_{GreedyGuess} * (1+\epsilon^{'}) \geq P_{opt}) \Rightarrow \epsilon < \epsilon^{'}\} \quad ,$$

where $\epsilon$ is the minimum weight that can guarantee the heuristic function to be admissible. Figure 1 shows that if we just keep $\epsilon$ constant, neglecting the changes of the estimate accuracy with the increase of the MAP variables, the estimate function and the optimal probability can be represented by the curve *Constant Weighting Heuristic*. Obviously, the problem with this idea is that it is less informed when the search progresses, as there are fewer MAP variables to estimate.

*Dynamic Weighting* (Pohl, 1973) is an efficient tool for improving the efficiency of A* search. If applied properly, it will keep the heuristic function admissible while remaining tight on the optimal probability. For MAP, in the shallow layer of the search tree, we get more MAP variables than the deeper layer for estimate. Hence, the greedy estimate will be more

likely to diverge from the optimal probability. We propose the following Dynamic Weighting Heuristic Function for the x*th* layer of the search tree of n MAP variables:

$$h(x) = GreedyGuess \cdot (1 + \alpha \frac{n-(x+1)}{n}) \quad (\alpha \geq \epsilon) .$$

Rather than keeping the weight constant throughout the search, we dynamically change it so as to make it less heavy as the search goes deeper. In the last step of the search ($x = n-1$), the weight will be zero, since the greedy guess for only one MAP variable is exact and then the cost function f(n-1) is equal to the probability of the assignment. Figure 1 shows an empirical comparison of greedy guess, constant, and dynamic weighting heuristics against accurate estimate of the probability. We see that the dynamic weighting heuristic is more informed than constant weighting.

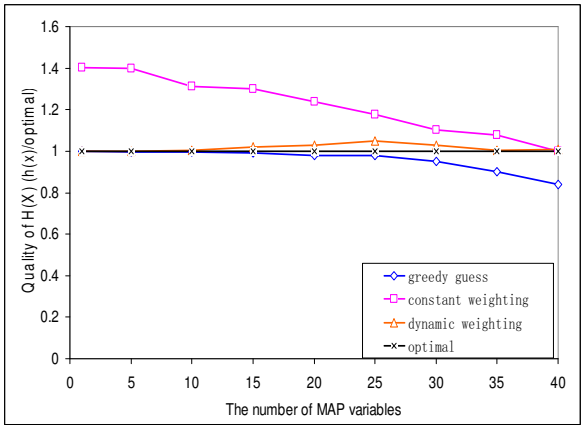



Figure 1: An empirical comparison of the constant weighting and the dynamic weighting heuristics based on greedy guess.

### 3.3 Searching with Inadmissible Heuristics for MAP Problem

Since the minimum weight $\epsilon$ that can guarantee the heuristic function to be admissible is unknown before the MAP problem is solved, and it may vary between cases, we normally set $\alpha$ to be a safe parameter that is supposed to be larger than $\epsilon$ (In our experiments, we set $\alpha$ to be 1.0). However, if $\alpha$ is accidentally smaller than $\epsilon$, it will lead the weighted heuristic to be inadmissible. Let us give this a closer look and analyze the conditions under which the algorithm fails to achieve optimality. Suppose there are two candidate assignments: $s_1$ and $s_2$ with probabilities $p_1$ and $p_2$ respectively, among which $s_2$ is the optimal assignment that the algorithm fails to find. And $s_1$ is now in the last step of search which will lead to a suboptimal solution. We skip the logarithm in the function for the sake of clarity here (then the cost function f is a product of transformed g and h instead of their sum).

$$f_1 = g_1 \cdot h_1 \text{ and } f_2 = g_2 \cdot h_2$$

The error introduced by a inadmissible $h_2$ is $f_1 > f_2$. The algorithm will then find $s_1$ instead of $s_2$, i.e.,

$$f_1 > f_2 \Rightarrow g_1 \cdot h_1 > g_2 \cdot h_2.$$

Since $s_1$ is now in the last step of search, $f_1 = p_1$ (Section 3.2.2). Now, suppose that we have an *ideal* heuristic function $h_2^{'}$, which leads to $p_2 = g_2 \cdot h_2^{'}$. Then we have:

$$\frac{g_1 \cdot h_1}{p_2} > \frac{g_2 \cdot h_2}{g_2 \cdot h_2^{'}} \Rightarrow \frac{p_1}{p_2} > \frac{g_2 \cdot h_2}{g_2 \cdot h_2^{'}} \Rightarrow \frac{p_1}{p_2} > \frac{h_2}{h_2^{'}} .$$

It is clear that only when the ratio between the probability of suboptimal assignment and the optimal one is larger than the ratio between the inadmissible heuristic function and the ideal one, will the algorithm find a suboptimal solution. Because of large asymmetries among probabilities that are further amplified by their multiplicative combination (Druzdzel, 1994), we can expect that for most cases, the ratios between $p_1$ and $p_2$ are far less than **1**. Even though the heuristic function will sometimes break the rule of admissibility, if only the greedy guess is not too divergent from the ideal estimate, the algorithm will still achieve optimality. Our simulation results also confirm the robustness of the algorithm.

### 3.4 Improvements to the Algorithm

There are two main techniques that we used to improve the efficiency of the basic A* algorithm.

### 3.4.1 Relevance Reasoning

The main problem faced by the decision-theoretic approach is the complexity of probabilistic reasoning. The critical factor in exact inference schemes for Bayesian networks is the topology of the underlying graph and, more specifically, its connectivity. The framework of relevance reasoning (Druzdzel and Suermondt (1994) provides an accessible summary of the relevant techniques) is based on *d*-separation and other simple and computational efficient techniques for pruning irrelevant parts of a Bayesian network and can yield subnetworks that are smaller and less densely connected than the original network. Relevance reasoning is an integral part of the SMILE library (Druzdzel, 2005) on which the implementation of our algorithm is based.

For MAP, our focus is the set of variables **M** and the evidence set **E**. Parts of the model that are probabilistically independent from the nodes in **M** given the observed evidence **E** are computationally irrelevant to reasoning about the MAP problem. Removing them leads to substantial savings in computation.

### 3.4.2 Dynamic Ordering

As the search tree is constructed dynamically, we have the freedom to order the variables in a way that will improve the efficiency of DWA*. Expanding nodes with the largest asymmetries in marginal probability distribution leads to early cut-off of less promising branches of the search tree. We use the entropy of the marginal probability distributions as a measure of asymmetry.

## 4 Experimental Results

To test DWA*, we compared its performance in real Bayesian networks with those of current state of the art MAP algorithms: the P-Loc and P-Sys algorithms (Park and Darwiche, 2001; Park and Darwiche, 2003) implemented in SamIam, and AnnealedMAP (Yuan et al., 2004) in SMILE respectively. We implemented DWA* in C++ and performed our tests on a 3.0 GHz Pentium D Windows XP computer with 2GB RAM. We used the default parameters and settings for all the three algorithms above during comparison, unless otherwise stated.

### 4.1 Experimental Design

The Bayesian networks that we used in our experiments included Alarm (Beinlich et al., 1989), Barley (Kristensen and Rasmussen, 2002), CPCS179 and CPCS360 (Pradhan et al., 1994), Diabetes (Andreassen et al., 1991), Hailfinder (Abramson et al., 1996), Munin (Andreassen et al., 1989), Pathfinder (Heckerman, 1990), Andes, and Win95pts (Heckerman et al., 1995). We also tested the algorithms on two large proprietary diagnostic networks built at the HRL Laboratories (HRL1 and HRL2). We divided the networks into three groups: (1) small and middle-sized, (2) large but tractable, and (3) hard networks.

For each network, we randomly generated 20 cases. For each case, we randomly chose 20 MAP variables from among the root nodes. We chose the same number of evidence nodes from among the leaf nodes. Following tests of MAP algorithms published earlier in the literature, we set the search time limit to be $3,000$ seconds.

### 4.2 Results of First & Second Group

We firstly ran the P-Loc, P-Sys, AnnealedMAP and DWA* on all networks in the first and second group. The P-Sys is an exact algorithm. So Table 2 only reports the number of MAP problems that were solved optimally by the rest three algorithms. DWA* found all optimal solutions. The P-Loc missed only one case on Andes and the AnnealedMAP missed one on Hailfinder and two cases on Andes.

Since both AnnealedMAP and P-Loc failed to find all optimal solutions in Andes, we studied the performance of the four algorithms as a function of the number of MAP variables (we randomly generated 20 cases for each number of MAP variables) on it.

Because P-Sys failed to generate any result when the number of MAP variables reached 40, while DWA* found all largest probabilities, we subsequently compared all the other three algorithms with DWA*. With the increase of the number of MAP variables, both P-Loc and

| Group | Networks | P-Loc | A-MAP | A* |
|---|---|---|---|---|
| 1 | Alarm | 20 | 20 | 20 |
| | CPCS179 | 20 | 20 | 20 |
| | CPCS360 | 20 | 20 | 20 |
| | Hailfinder | 20 | 19 | 20 |
| | Pathfinder | 20 | 20 | 20 |
| | Andes | 19 | 18 | 20 |
| | Win95pts | 20 | 20 | 20 |
| 2 | Munin | 20 | 20 | 20 |
| | HRL1 | 20 | 20 | 20 |
| | HRL2 | 20 | 20 | 20 |

Table 1: Number of cases solved optimally out of 20 cases for the first and second group.

| #MAP | P-Sys | P-Loc | A-MAP |
|---|---|---|---|
| 10 | 0 | 0 | 0 |
| 20 | 0 | 1 | 2 |
| 30 | 0 | 1 | 0 |
| 40 | TimeOut | 4 | 4 |
| 50 | TimeOut | 6 | 2 |
| 60 | TimeOut | 5 | 2 |
| 70 | TimeOut | 6 | 5 |
| 80 | TimeOut | 6 | 1 |

Table 2: Number of cases in which DWA* found more probable instantiation than the other three algorithms (network Andes).

AnnealedMAP turned out to be less accurate than DWA* on Andes. When the number of MAP variables was above 40, there were about 25% cases of P-Loc and 15% cases in which AnnealedMAP found smaller probabilities than DWA*. We notice from Table 2 that P-Loc spent less time than DWA* when using its default settings for Andes, so we increased the search steps of P-Loc such that it spent the same amount of time as DWA* in order to make a fair comparison. However, in practice the search time is not continuous in the number of search steps, so we just chose parameters for P-Loc such that it spent only a little bit more time than DWA*. Table 3 shows the comparison results. We can see that after increasing the search steps of P-Loc, DWA* still maintains better accuracy.

In addition to the precision of the results, we also compared the efficiency of the algorithms. Table 4 reports the average running time of the four algorithms on the first and the second groups of networks. For the first group, the AnnealedMAP, P-Loc and P-Sys algorithms showed similar efficiency on all except the CPCS360 and Andes networks. DWA* generated solutions with the shortest time on average. Its smaller variance of the search time indicates that DWA* is more stable across different networks.

| #MAP | P-Loc<DWA* | P-Loc>DWA* |
|---|---|---|
| 10 | 0 | 0 |
| 20 | 0 | 0 |
| 30 | 0 | 0 |
| 40 | 1 | 0 |
| 50 | 2 | 0 |
| 60 | 2 | 1 |
| 70 | 3 | 2 |
| 80 | 5 | 0 |

Table 3: The number of cases in which the P-Loc algorithm found larger/smaller probabilities than DWA* in network Andes when spending a little bit more time than DWA*.

| | P-Sys | P-Loc | A-MAP | A* |
|---|---|---|---|---|
| Alarm | 0.017 | 0.020 | 0.042 | 0.005 |
| CPCS179 | 0.031 | 0.117 | 0.257 | 0.024 |
| CPCS360 | 0.045 | 75.20 | 0.427 | 0.072 |
| Hailfinder | 2.281 | 0.109 | 0.219 | 0.266 |
| Pathfinder | 0.052 | 0.056 | 0.098 | 0.005 |
| Andes | 14.49 | 1.250 | 4.283 | 2.406 |
| Win95pts | 0.035 | 0.041 | 0.328 | 0.032 |
| Munin | 3.064 | 4.101 | 19.24 | 1.763 |
| HRL1 | 0.493 | 51.18 | 2.831 | 0.193 |
| HRL2 | 0.092 | 3.011 | 2.041 | 0.169 |

Table 4: Running time (in seconds) of the four algorithms on the first and second group.

For the second group, which consists of large Bayesian networks, P-Sys, AnnealedMAP and DWA* are all efficient. DWA* still spends shortest time on average, while the P-Loc is much slower on the HRL1 network.

### 4.3 Results of Third Group

The third group consists of two complex Bayesian networks: Barley and Diabetes, many nodes of which have more than 10 different states. Because the P-Sys algorithm did not produce results within the time limit, the only available measure of accuracy was a relative one: which of the algorithms found an assignment with higher probability. Table 5 lists the number of cases that were solved differently between the P-Loc, AnnealedMAP, and DWA* algorithms. $P_L$, $P_A$ and $P_*$ stand for the probability of MAP solutions found by P-Loc, AnnealedMAP and DWA* respectively.

| Group3 | $P_*>P_L/P_*<P_L$ | $P_*>P_A/P_*<P_A$ |
|---|---|---|
| Barley | 3/2 | 5/3 |
| Diabetes | 5/0 | 4/0 |

Table 5: The numberof cases that are solved differently from P-Loc, AnnealedMAP and DWA*.

For Barley, the accuracy of the three algorithms was quite similar. However, for Diabetes DWA* was more accurate: it found solutions with largest probabilities for all 20 cases, while P-Loc failed to find 5 and AnnealedMAP failed to find 4 of them.

| Group3 | P-Sys | P-Loc | A-MAP | A* |
|---|---|---|---|---|
| Barley | TimeOut | 68.63 | 31.95 | 122.1 |
| Diabetes | TimeOut | 338.4 | 163.4 | 81.8 |

Table 6: Running time (in seconds) of the four algorithms on the third group.

DWA* turns out to be slower than P-Loc and AnnealedMAP on Barley but more efficient on Diabetes (see Table 6).

### 4.4 Results of Incremental MAP Test

Out last experiment focused on the robustness of the four algorithms to the number of MAP variables. In this experiment, we set the number of evidence nodes to 100, generated MAP problems with an increasing number of MAP nodes. We chose the Munin network, because it seems the hardest network among the group 1 & 2 and has sufficiently large sets of root and leaf nodes. The running times are shown in Figure 2. Typically, P-Sys and P-Loc need more running time in face of more complex problems, while AnnealedMAP and DWA* seem more robust in comparison.

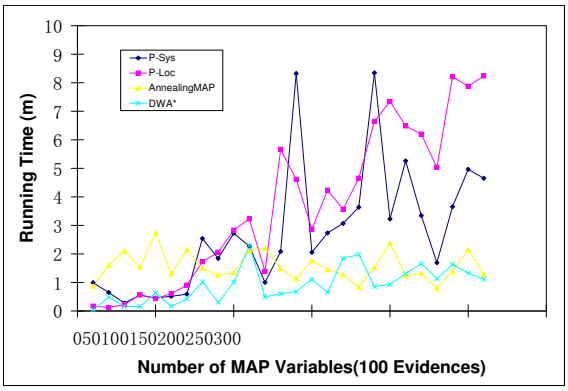



Figure 2: Running time of the four algorithms when increasing the number of MAP nodes on the Munin network.

## 5 Discussion

Solving MAP is hard. By exploiting asymmetries among the probabilities of possible elements of the joint probability distributions of MAP variables, DWA* is able to greatly reduce the search space and lead to efficient and accurate solutions of MAP problems. Our experimental results also show that generally, DWA* is more efficient than the existent algorithms. Especially for large and complex Bayesian networks, when the exact algorithm fails to generate any result within a reasonable time, the DWA* can still provide accurate solutions efficiently. Further extension of this research is to apply DWA* to the K-MAP problem, which is to find k most probable assignments for MAP variables. It is very convenient for DWA* to achieve that, since in the process of finding the most probable assignment the algorithm keeps all the candidate assignments in the search frontier. We can expect that the additional search time will be linear in k.

In sum, DWA* enriches the approaches for solving MAP problem and extends the scope of MAP problems that can be solved.

**Acknowledgements**

This research was supported by the Air Force Office of Scientific Research grants F49620–03–1–0187 and FA9550–06–1–0243 and by Intel Research. We thank anonymous reviewers for several insightful comments that led to improvements in the presentation of the paper. All experimental data have been obtained using SMILE, a Bayesian inference engine developed at the Decision Systems Laboratory and available at `http://genie.sis.pitt.edu/`.

## References

B. Abramson, J. Brown, W. Edwards, A. Murphy, and R. Winkler. 1996. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12(1):57–72.

S. Andreassen, F. V. Jensen, S. K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A. R. Sørensen, A. Rosenfalck, and F. Jensen. 1989. MUNIN — an expert EMG assistant. In John E. Desmedt, editor, *Computer-Aided Electromyography and Expert Systems*, chapter 21. Elsevier Science Publishers, Amsterdam.

S. Andreassen, R. Hovorka, J. Benn, K. G. Olesen, and E. R. Carson. 1991. A model-based approach to insulin adjustment. In M. Stefanelli, A. Hasman, M. Fieschi, and J. Talmon, editors, *Proceedings of the Third Conference on Artificial Intelligence in Medicine*, pages 239–248. Springer-Verlag.

I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. 1989. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *In Proc. 2'nd European Conf. on AI and Medicine*, pages 38:247–256, Springer-Verlag, Berlin.

Marek J. Druzdzel and Henri J. Suermondt. 1994. Relevance in probabilistic models: "Backyards" in a "small world". In *Working notes of the AAAI–1994 Fall Symposium Series: Relevance*, pages 60–63, New Orleans, LA (An extended version of this paper is in preparation.), 4–6 November.

M. J. Druzdzel. 1994. Some properties of joint probability distributions. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI–94)*, pages 187–194, Morgan Kaufmann Publishers San Francisco, California.

M. J. Druzdzel. 2005. Intelligent decision support systems based on smile. *Software 2.0*, 2:12–33.

D. Heckerman, J. Breese, and K. Rommelse. 1995. Decision-theoretic troubleshooting. *Communications of the ACM*, 38:49–57.

D. Heckerman. 1990. Probabilistic similarity networks. *Networks*, 20(5):607–636, August.

K. Kristensen and I.A. Rasmussen. 2002. The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture*, 33:197–217.

J. D. Park and A. Darwiche. 2001. Approximating MAP using local search. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI–01)*, pages 403–410, Morgan Kaufmann Publishers San Francisco, California.

J. D. Park and A. Darwiche. 2003. Solving MAP exactly using systematic search. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI–03)*, pages 459–468, Morgan Kaufmann Publishers San Francisco, California.

J. D. Park. 2002. MAP complexity results and approximation methods. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI–02)*, pages 388–396, Morgan Kaufmann Publishers San Francisco, California.

J. Pearl. 1988. *Heuristics : intelligent search strategies for computer problem solving.* Addison-Wesley Publishing Company, Inc.

I. Pohl. 1973. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *Proc. of the 3rd IJCAI*, pages 12–17, Stanford, MA.

M. Pradhan, G. Provan, B. Middleton, and M. Henrion. 1994. Knowledge engineering for large belief networks. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)*, pages 484–490, San Mateo, CA. Morgan Kaufmann Publishers, Inc.

S. E. Shimony. 1994. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68:399–410.

C. Yuan, T. Lu, and M. J. Druzdzel. 2004. Annealed MAP. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI–04)*, pages 628–635, AUAI Press, Arlington, Virginia.

# A Short Note on Discrete Representability of Independence Models

Petr Šimeček
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 4,
182 08 Prague, Czech Republic

**Abstract**

The paper discusses the problem to characterize collections of conditional independence triples (independence model) that are representable by a discrete distribution. The known results are summarized and the number of representable models over four elements set, often mistakenly claimed to be 18300, is corrected. In the second part, the bounds for the number of positively representable models over four elements set are derived.

## 1 Introduction

Conditional independence relationships occur naturally among components of highly structured stochastic systems. In a field of graphical Markov models, a graph (nodes connected by edges) is used to represent the CI structure of a set of probability distributions.

Given the joint probability distribution of a collection of random variables it is easy to construct the list of all CIs among them. On the other hand, given the list (here called an independence model) of CIs an interesting question arises whether there exists a collection of discrete random variables meeting these and only these CIs, i.e. representing that model.

The problem of probabilistic representability comes originally from J. Pearl, cf. (Pearl, 1998). It was proved by M. Studený in (Studený, 1992) that there is no finite characterization (≡finite set of inference rules) of the set of representable independence models.

The interesting point is that for the fixed number of variables (or vertices) the number of representable independence models is much higher than the number of graphs. Therefore, even partial characterization of representable independence models may help to improve and understand the limits of learning of Bayesian networks and Markov models.

## 2 Independence models

For the reader's convenience, auxiliary results related to independence models are recalled in this section.

Throughout the paper, the singleton {a} will be shorten by $a$ and the union of sets $A \cup B$ will be written simply as juxtaposition $AB$. A random vector $\boldsymbol{\xi} = (\xi_a)_{a\in N}$ is a collection of random variables indexed by a finite set $N$. For $A \subseteq N$, a subvector $(\xi_a)_{a\in A}$ is denoted by $\boldsymbol{\xi}_A$; $\boldsymbol{\xi}_\emptyset$ is presumed to be a constant. Analogously, if $\boldsymbol{x} = (x_a)_{a\in N}$ is a constant vector then $\boldsymbol{x}_A$ is an appropriate coordinate projection.

Provided $A, B, C$ are pairwise disjoint subsets of $N$, "$\boldsymbol{\xi}_A \perp\!\!\!\perp \boldsymbol{\xi}_B | \boldsymbol{\xi}_C$" stands for a statement $\boldsymbol{\xi}_A$ and $\boldsymbol{\xi}_B$ are conditionally independent given $\boldsymbol{\xi}_C$. In particular, unconditional independence ($C = \emptyset$) is abbreviated as $\boldsymbol{\xi}_A \perp\!\!\!\perp \boldsymbol{\xi}_B$.

A random vector $\boldsymbol{\xi} = (\xi_a)_{a\in N}$ is called **discrete** if each $\xi_a$ takes values in a state space $X_a$ such that $1 < |X_a| < \infty$. A discrete random vector $\boldsymbol{\xi}$ is called **positive** if for any appropriate constant vector $\boldsymbol{x}$

$$0 < P(\boldsymbol{\xi} = \boldsymbol{x}) < 1.$$

In the case of discretely distributed random vector, variables $\xi_a$ and $\xi_b$ are independent[1] given

[1]The independence relation between random vectors

$\boldsymbol{\xi}_C$ iff for any appropriate constant vector $\boldsymbol{x}_{abC}$

$$P(\boldsymbol{\xi}_{abC} = \boldsymbol{x}_{abC})P(\boldsymbol{\xi}_C = \boldsymbol{x}_C) \quad = \\ P(\boldsymbol{\xi}_{aC} = \boldsymbol{x}_{aC})P(\boldsymbol{\xi}_{bC} = \boldsymbol{x}_{bC}).$$

Let $N$ be a finite set and $\mathcal{T}_N$ denotes the set of all pairs $\langle ab|C\rangle$ such that $ab$ is an (unordered) couple of distinct elements of $N$ and $C \subseteq N \setminus ab$.

Subsets of $\mathcal{T}_N$ will be referred as formal **independence models** over $N$. Independence models $\emptyset$ and $\mathcal{T}_N$ are called trivial.

The independence model $\mathcal{I}(\boldsymbol{\xi})$ induced by a random vector $\boldsymbol{\xi}$ indexed by $N$ is the independence model over $N$ defined as follows

$$\mathcal{I}(\boldsymbol{\xi}) = \{\langle ab|C\rangle;\ \xi_a \perp\!\!\!\perp \xi_b | \boldsymbol{\xi}_C\}\,.$$

Let us emphasize that an independence model $\mathcal{I}(\boldsymbol{\xi})$ uniquely determines also all other conditional independences among subvectors of $\boldsymbol{\xi}$, cf. (Matúš, 1992).

Diagrams proposed by R. Lněnička will be used for a visualisation of independence model $I$ over $N$ such that $|N| \leq 4$. Each element of $N$ is plotted as a dot. If $\langle ab|\emptyset\rangle \in I$ then dots corresponding to $a$ and $b$ are joined by a line. If $\langle ab|c\rangle \in I$ then we put a line between dots corresponding to $a$ and $b$ and add small line in the middle pointing in $c$–direction. If both $\langle ab|c\rangle$ and $\langle ab|d\rangle$ are elements of $I$, then only one line with two small lines in the middle is plotted. Finally, if $\langle ab|cd\rangle \in I$ is visualised by a brace between $a$ and $b$. See example in Figure 1.

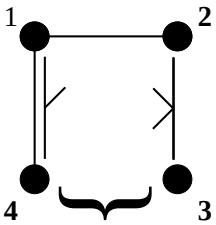



Figure 1: Diagram of the independence model $I = \big\{\langle 12|\emptyset\rangle,\ \langle 23|1\rangle,\ \langle 23|4\rangle,\ \langle 34|12\rangle,\ \langle 14|\emptyset\rangle,\ \langle 14|2\rangle\big\}$.

Independence models $I$ and $I^*$ over $N$ will be called **isomorphic** if there exists a permutation $\pi$ on $N$ such that

$$\langle ab|C\rangle \in I \Longleftrightarrow \langle \pi(a)\pi(b)|\pi(C)\rangle \in I^*,$$

where $\pi(C)$ stands for $\{\pi(c);\ c \in C\}$. See Figure 2 for an example of three isomorphic models.

An equivalence class of independence models with respect to the isomorphic relation will be referred as **type**.

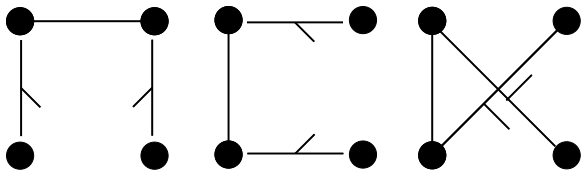


Figure 2: Example of three isomorphic models.

An independence model $I$ is said to be **representable**[2] if there exists a discretely distributed random vector $\boldsymbol{\xi}$ such that $I = \mathcal{I}(\boldsymbol{\xi})$. In addition, a special attention will be devoted to **positive representations**, i.e. representations by a positive discrete distribution.

Let us note that isomorphic models are either all representable or non-representable. Consequently, we can classify types as representable and non-representable.

**Lemma 1.** *If $I = \mathcal{I}(\boldsymbol{\xi})$ and $I^* = \mathcal{I}(\boldsymbol{\xi}^*)$ are representable independence models then the independence model $I \cap I^*$ is also representable. In particular, if they have positive representations then there exists a positive representation of $I \cap I^*$, too.*

*Proof.* Let $X = \prod X_a$ and $X' = \prod X'_a$ be state spaces of $\boldsymbol{\xi}$ and $\boldsymbol{\xi}'$, respectively. The required representation $\hat{\boldsymbol{\xi}}$ takes place in

$$\widehat{X} = \prod_{a \in N} X_a \times X'_a$$

and it is distributed as follows

$$P(\hat{\boldsymbol{\xi}} = (x_a, x'_a)_{a\in N}) = \\ P(\boldsymbol{\xi} = (x_a)_{a\in N}) \cdot P(\boldsymbol{\xi}' = (x'_a)_{a\in N}).$$

See (Studený and Vejnarová, 1998), pp. 5, for more details. □

$\boldsymbol{\xi}_A, \boldsymbol{\xi}_B$ given $\boldsymbol{\xi}_C$ might be defined analogously. However, we will see that such relationships are uniquely determined by the elementary ones ($|A| = |B| = 1$).

[2] Of course, it is also possible to consider representability in other distribution frameworks that the discrete distributions, cf. (Lněnička, 2005).

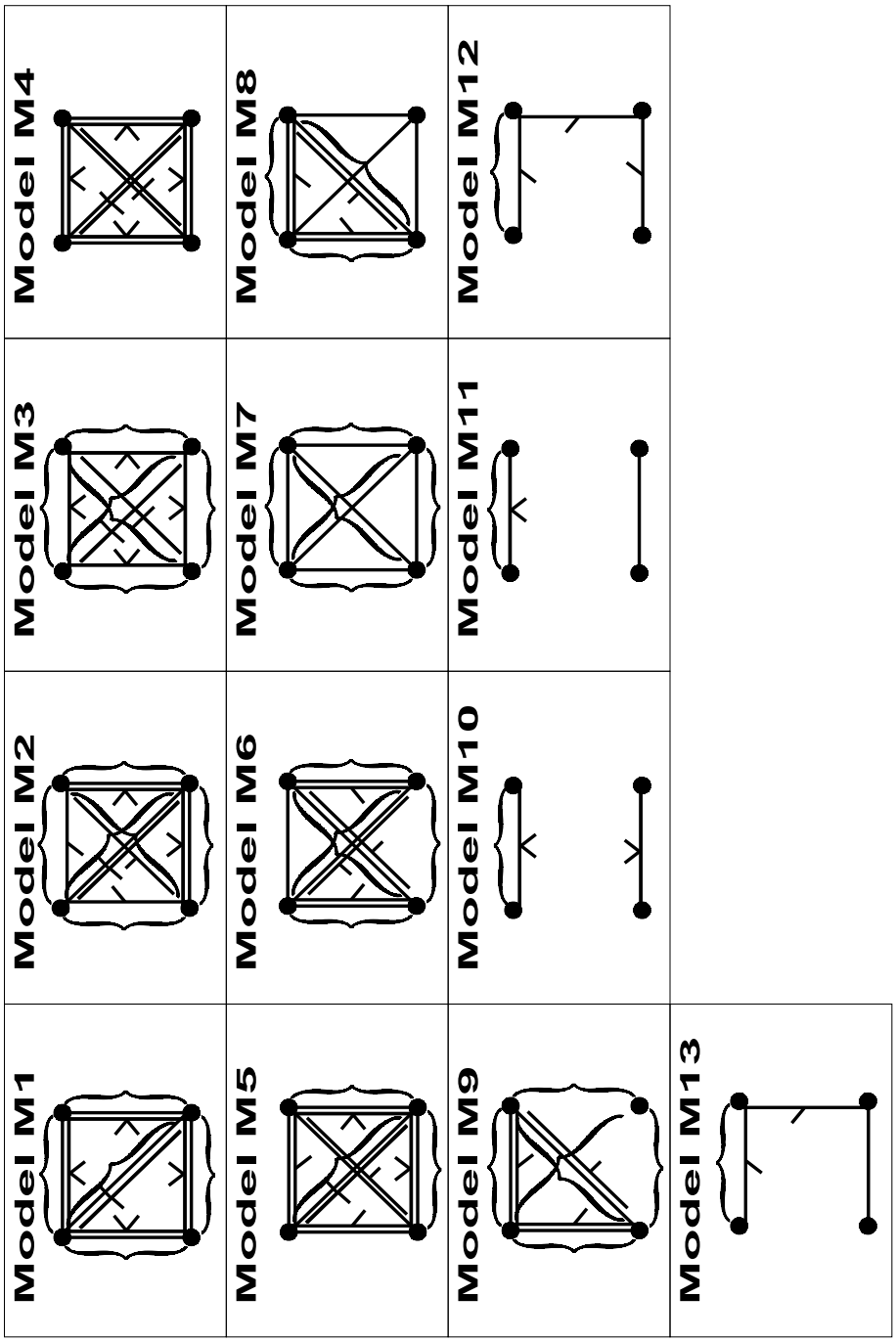



Figure 3: Irreducible models over $N = \{1, 2, 3, 4\}$.

**Lemma 2.** *Let $a, b, c$ be distinct elements of $N$ and $D \subseteq N \setminus abc$. If an independence models $I$ over $N$ is representable, then*

$$\big(\{\langle ab|cD\rangle, \langle ac|D\rangle\} \subseteq I\big) \Longleftrightarrow \big(\{\langle ac|bD\rangle, \langle ab|D\rangle\} \subseteq I\big).$$

*Moreover, if $I$ is positively representable, then*

$$\big(\{\langle ab|cD\rangle, \langle ac|bD\rangle\} \subseteq I\big) \Longrightarrow \big(\{\langle ab|D\rangle, \langle ac|D\rangle\} \subseteq I\big).$$

*Proof.* These are so called "semigraphoid" and "graphoid" properties, cf. (Lauritzen, 1996) for the proof. □

## 3 Representability of Independence Models over $N = \{1, 2, 3, 4\}$

For $N$ consisting of three or less elements, all independence models not contradicting properties from Lemma 2 are representable, resp. positively representable, cf. (Studený, 2005). That is why we focus on $N = \{1, 2, 3, 4\}$ from now to the end of the paper.

The first subsection summarizes known results related to (general) representability of independence models. The second subsection is devoted to positive representability.

### 3.1 General Representability

The problem was solved in a brilliant series of papers (Matúš and Studený, 1995), (Matúš, 1995) and (Matúš, 1999) by F. Matúš and M. Studený. The final conclusions are clearly and comprehensibly presented in (Studený and Boček, 1994) and (Matúš, 1997)[3].

In brief, due to Lemma 1 an intersection of two representable models is also representable. Therefore, the class of all representable models over $N$ can be described by a set of so called **irreducible** models $\mathcal{C}$, i.e. nontrivial representable models that cannot be written as an intersection of two other representable models. It is not difficult to evidence that a nontrivial independence model $I$ is representable if and only if there exists $\mathcal{A} \subseteq \mathcal{C}$ such that

$$I = \bigcap_{C \in \mathcal{A}} C.$$

There are only only 13 types of irreducible models, see Figure 3 or (Studený and Boček, 1994), pp. 277–278. The problematic point is the total number of representable independence models over $N$. It has been believed that this number is 18300, cf. (Studený, 2002), (Lauritzen and Richardson, 2002), (Robins et al., 2003), (Šimeček, 2006)... However, working on this paper I have discovered that there actually exist 18478 different representable independence models over $N$ of 1098 types. The list of models has been checked by several programs including SG POKUS written by M. Studený and P. Boček. The list can be downloaded from the web page

`http://5r.matfyz.cz/skola/models`

### 3.2 Positive Representability

Only a little is known about positive representability. This paper would like to be the first step to the complete characterisation of positively representable models over $N = \{1, 2, 3, 4\}$.

Obviously, a set of positively representable models is a subset of the set of (generally) representable models. In addition, positively representable model must fulfill properties following the second part of Lemma 2 and Lemma 3 below.

**Lemma 3.** *Let $a, b, c, d$ be distinct elements of $N$. If $I$ is a positively representable independence model oven $N$ such that*

$$\{\langle ab|cd\rangle, \langle cd|ab\rangle, \langle cd|a\rangle\} \subseteq I,$$

*then*

$$\langle cd|b\rangle \in I \Longleftrightarrow \langle cd|\emptyset\rangle \in I.$$

*Proof.* See (Spohn, 1994), pp. 15. □

There are 5547 (generally) representable models (356 types) meeting requirements on for positively representable models from Lemma 2 and Lemma 3. This is the upper bound to the set of all positively representable models.

[3] To avoid confusion, note that (Matúš, 1997) contains a minor typo in Figure 14 on pp. 21. Over the upper line in the first two diagrams should be $\emptyset$ instead of $*$.

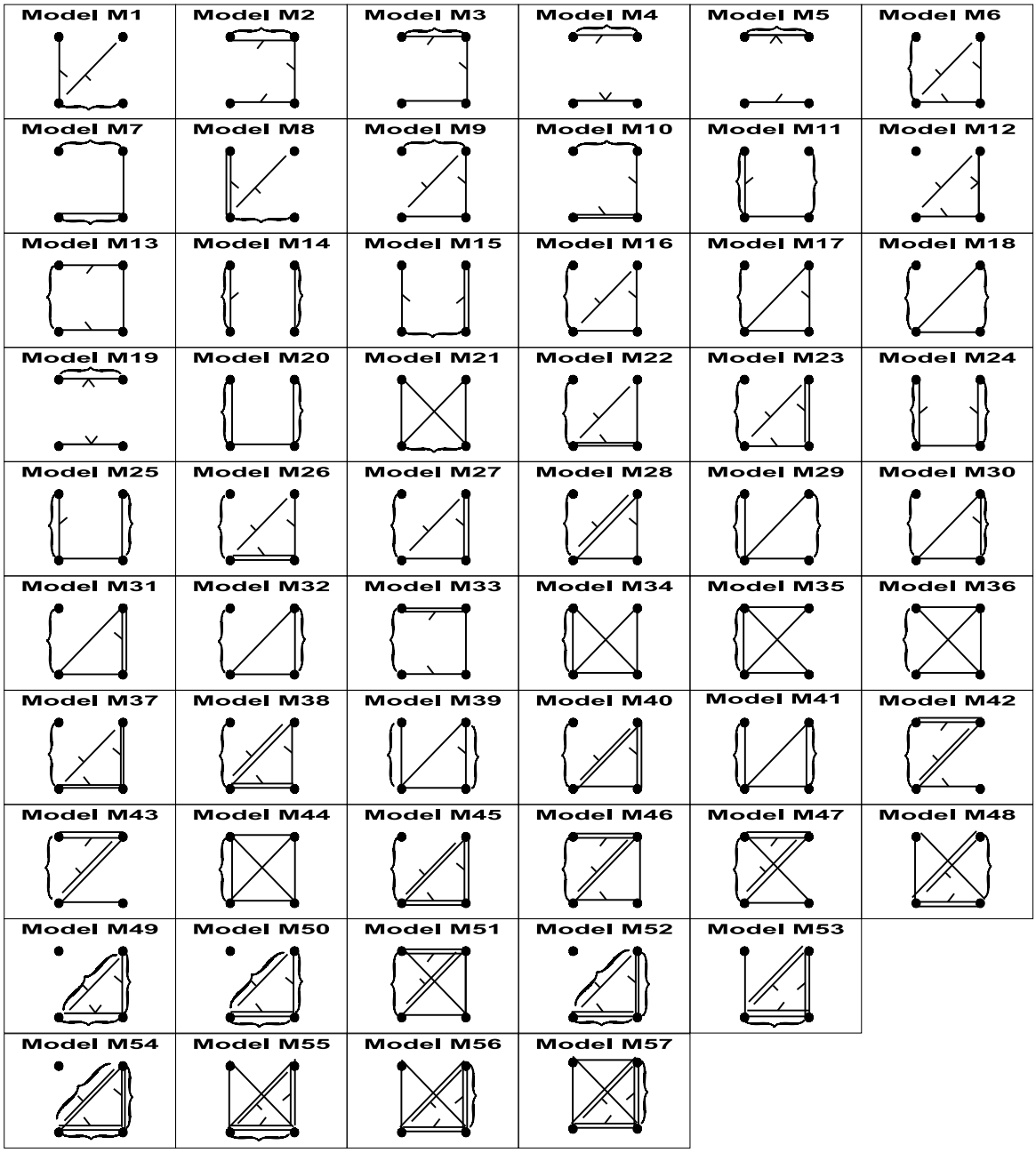



Figure 4: Undecided types.

Again, this class of models $\mathcal{A}$ is generated by its subset $\mathcal{C}$ by the operation of intersection. The elements of $\mathcal{C}$ can be found by starting with an empty set $\mathcal{C}$ and in each step adding the element of $\mathcal{A}$ not yet generated by $\mathcal{C}$ with the greatest size. Actually, $\mathcal{C}$ contains (nontrivial) models of 23 types (may be downloaded from the mentioned web page).

To find some lower bound to the set of positively representable models, large amount of positive binary (≡sample space $\{0,1\}^N$) random distributions have been randomly generated. The description of the generating process will be omitted here[4], see the above mentioned web page for the list of corresponding independence models and their binary representations. Using Lemma 1 we obtained 4555 (299 types) different positive representations of independence models. The remaining problematic 57 types are plotted in Figure 4.

### 3.3 Conclusion

Let us summarize the results into the concluding theorem.

**Theorem 1.** *There are* 18478 *different (generally) representable independence models (*1098 *types) over the set* $N = \{1,2,3,4\}$. *There are between* 4555 *and* 5547 *different positively representable independence models (*299–356 *types) over the set* $N = \{1,2,3,4\}$.


### Acknowledgments

This work was supported by the grants GA ČR n. 201/04/0393 and GA ČR n. 201/05/H007.


## References


S. L. Lauritzen and T. S. Richardson. 2002. Chain graph models and their causal interpretations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64:321–361.

S. L. Lauritzen. 1996. *Graphical Models*. Oxford University Press.

R. Lněnička. 2005. On gaussian conditional independence structures. Technical report, ÚTIA AV ČR.

F. Matúš and M. Studený. 1995. Conditional independences among four random variables I. *Combinatorics, Probability & Computing*, 4:269–278.

F. Matúš. 1992. On equivalence of markov properties over undirected graphs. *Journal of Applied Probability*, 29:745–749.

F. Matúš. 1995. Conditional independences among four random variables II. *Combinatorics, Probability & Computing*, 4:407–417.

F. Matúš. 1997. Conditional independence structures examined via minors. *The Annals of Mathematics and Artificial Intelligence*, 21:99–128.

F. Matúš. 1999. Conditional independences among four random variables III. *Combinatorics, Probability & Computing*, 8:269–276.

J. Pearl. 1998. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

J. M. Robins, R. Scheines, P. Spirtes, and L. Wasserman. 2003. Uniform consistency in causal inference. *Biometrika*, 90:491–515.

P. Šimeček. 2006. Gaussian representation of independence models over four random variables. Accepted to COMPSTAT conference.

W. Spohn. 1994. On the properties of conditional independence. In P. Humphreys, editor, *Patrick Suppes: Scientific Philosopher*, volume 1, pages 173–194. Kluwer.

M. Studený and P. Boček. 1994. CI-models arising among 4 random variables. In *Proceedings of the 3rd workshop WUPES*, pages 268–282.

M. Studený and J. Vejnarová. 1998. The multiinformation function as a tool for measuring stochastic dependence. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 261–298. Kluwer.

M. Studený. 1992. Conditional independence relations have no finite complete characterization. In S. Kubík and J.A. Víšek, editors, *Information Theory, Statistical Decision Functions and Random Processes. Transactions of the 11th Prague Conference*, volume B, pages 377–396. Kluwer.

M. Studený. 2002. On stochastic conditional independence: the problems of characterization and description. *Annals of Mathematics and Artificial Intelligence*, 35:241–323.

M. Studený. 2005. *Probabilistic Conditional Independence Structures*. Springer.


[4]Briefly, the parametrization of joint distribution by moments has been used. Further, both systematic and random search through parameter space was performed.

# Evaluating Causal effects using Chain Event Graphs

Peter Thwaites and Jim Smith
University of Warwick Statistics Department
Coventry, United Kingdom

**Abstract**

The Chain Event Graph (CEG) is a coloured mixed graph used for the representation of finite discrete distributions. It can be derived from an Event Tree (ET) together with a set of equivalence statements relating to the probabilistic structure of the ET. CEGs are especially useful for representing and analysing asymmetric processes, and collections of implied conditional independence statements over a variety of functions can be read from their topology. The CEG is also a valuable framework for expressing causal hypotheses, and manipulated-probability expressions analogous to that given by Pearl in his Back Door Theorem can be derived. The expression we derive here is valid for a far larger set of interventions than can be analysed using Bayesian Networks (BNs), and also for models which have insufficient symmetry to be described adequately by a Bayesian Network.

## 1 Introduction

Bayesian Networks are good graphical representations for many discrete joint probability distributions. However, many asymmetric models (by which we mean models with non-symmetric sample space structures) cannot be fully described by a BN. Such processes arise frequently in, for example, biological regulation, risk analysis and Bayesian policy analysis.

In eliciting these models it is usually sensible to start with an Event Tree (Shafer, 1996), which is essentially a description of how the process unfolds rather than how the system might appear to an observer. Working with an ET can be quite cumbersome, but they do reflect any model asymmetry, both in model development and in model sample space structure.

The Chain Event Graph (Riccomagno & Smith, 2005; Smith & Anderson, 2006; Thwaites & Smith, 2006a) is a graphical structure designed for analysis of asymmetric systems. It retains the advantages of the ET, whilst typically having far fewer edges and vertices. Moreover, the CEG can be read for a rich collection of conditional independence properties of the model. Unlike Jaeger's very useful Probabilistic Decision Graph (2002) this includes all the properties that can be read from the equivalent BN if the CEG represents a symmetric model and far more if the model is asymmetric.

In the next section we show how CEGs can be constructed. We then describe how we can use CEGs to analyse the effects of Causal manipulation.

Bayesian Networks are often extended to apply also to a control space. When it is valid to make this extension the BN is called causal. Although there is debate (Pearl, 2000; Lauritzen, 2001; Dawid, 2002) about terminology, it is certainly the case that BNs are useful for analysing (in Pearl's notation) the effects of manipulations of the form $Do\ X = x_0$ in symmetric models, where $X$ is a variable to be manipulated and $x_0$ the setting that this variable is to be manipulated to. This type of intervention, which might be termed atomic, is actually a rather coarse manipulation since we would need to extend the space to make predictions of effects when $X$ is manipulated to **any** value. Although there is a case for only considering such manipulations when a model is very symmetric, it is too coarse to capture many of the manipulations we might want to consider in asymmetric environments. We use this paper to show how CEGs can be used to analyse a far

more refined *singular* manipulation in models which may have insufficient symmetry to be described adequately by a Bayesian Network.

## 2 CEG construction

We can produce a CEG from an Event Tree which we believe represents the model (see for example Figure 1). This ET is just a graphical description of how the process unfolds, and the set of atoms of the Event Space (or path sigma algebra) of the tree is simply the set of root to leaf paths within the tree. Any random variables defined on the tree are measureable with respect to this path sigma algebra.

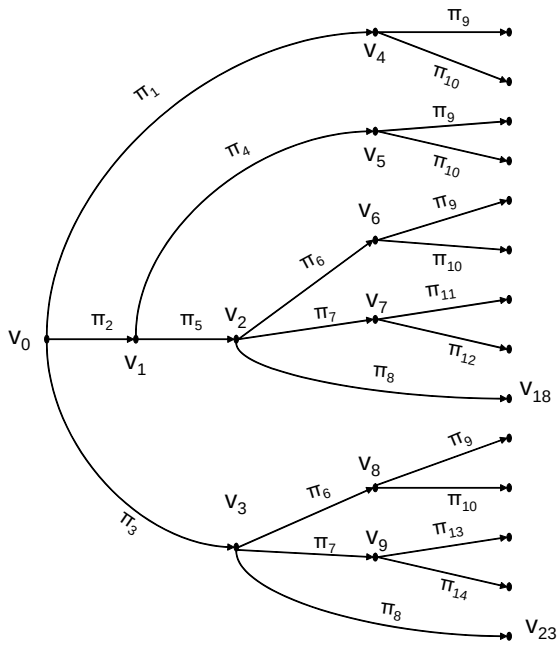



Figure 1. ET for machine example.

**Example 1.**
A machine in a production line utilises two replaceable components A and B. Faults in these components do not automatically cause the machine to fail, but do affect the quality of the product, so the machine incorporates an automated monitoring system, which is completely reliable for finding faults in A, but which can detect a fault in B when it is functioning correctly.

In any monitoring cycle, component A is checked first, and there are three initial possibilities: A, B checked and no faults found ($\pi_1$ on the ET in Figure 1); A checked, fault found, machine switched off ($\pi_2$); A checked, no fault found, B checked, fault found, machine switched off ($\pi_3$).

If A is found faulty it is replaced and the machine switched back on (vertex $v_1$), and B is then checked. B is then either found not faulty ($\pi_4$), or faulty and the machine switched off ($\pi_5$).

If B is found faulty by the monitoring system, then it is removed and checked (vertices $v_2$ and $v_3$). There are then three possibilities, whose probabilities are independent of whether or not component A has been replaced: B is not in fact faulty, the machine is reset and restarted ($\pi_6$); B is faulty, is successfully replaced and the machine restarted ($\pi_7$); B is faulty, is replaced unsuccessfully and the machine is left off until the engineer can see it ($\pi_8$).

At the time of any monitoring cycle, the quality of the product produced ($\pi_{10}$) is unaffected by the replacement of A unless B is also replaced. It is however dependent on the *effectiveness* of B which depends on its *age*, but also, if it is a **new** component, on the *age* of A; so:
$\pi(\textit{good product} \mid \textit{A and B replaced}) = \pi_{12}$
$> \pi(\textit{good product} \mid \textit{only B replaced}) = \pi_{14}$
$> \pi(\textit{good product} \mid \textit{B not replaced}) = \pi_{10}$

An ET for this set-up is given in Figure 1 and a derived CEG in Figure 2. Note that:

- The subtrees rooted in the vertices $v_4$, $v_5$, $v_6$ and $v_8$ of the ET are identical (both in physical structure and in probability distribution), so these vertices have been conjoined into the vertex (or *position*) $w_4$ in the CEG.
- The subtrees rooted in $v_2$ and $v_3$ are not identical (as $\pi_{11} \neq \pi_{13}$, $\pi_{12} \neq \pi_{14}$), but the edges leaving $v_2$ and $v_3$ carry identical probabilities. The equivalent *positions* in the CEG $w_2$ and $w_3$ have been joined by an undirected edge.
- All leaf-vertices of the ET have been conjoined into one sink-vertex in the CEG, labelled $w_\infty$.

To complete the transformation, note that $v_i \to w_i$ for $0 \leq i \leq 3$, $v_7 \to w_5$ and $v_9 \to w_6$.

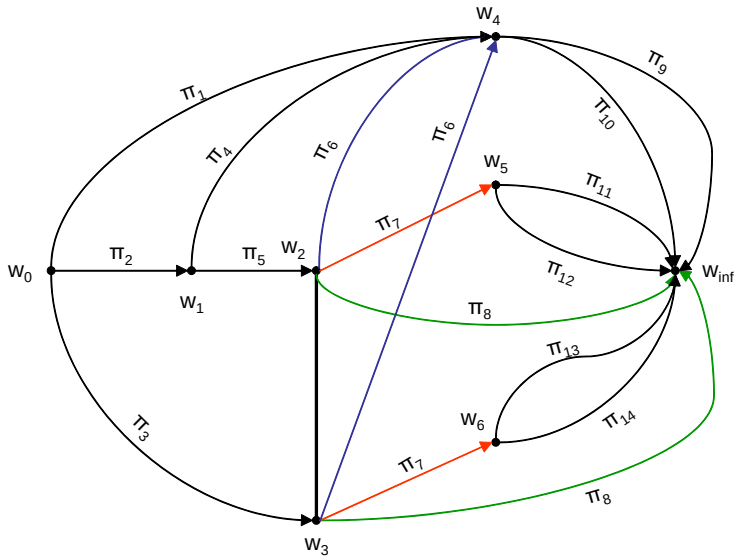



Figure 2. CEG for machine example.

A formal description of the process is as follows: Consider the ET $T = (V(T), E(T))$ where each element of $E(T)$ has an associated edge probability. Let $S(T) \subset V(T)$ be the set of non-leaf vertices of the ET.
Let $v_i \prec v_j$ indicate that there is a path joining vertices $v_i$ and $v_j$ in the ET, and that $v_i$ precedes $v_j$ on this path.
Let $\mathbb{X}(v)$ be the sample space of $X(v)$, the random variable associated with the vertex $v$ ($\mathbb{X}(v)$ can be thought of as the set of edges leaving $v$, so in our example, $\mathbb{X}(v_1) = \{B\ found\ not\ faulty, B\ found\ faulty\}$).

For any $v \in S(T)$, $v_l \in V(T)\backslash S(T)$ such that $v \prec v_l$ :

- Label $v = {v_\lambda}^0$
- Let ${v_\lambda}^{i+1}$ be the vertex such that ${v_\lambda}^i \prec {v_\lambda}^{i+1} \prec v_l$ for which there is no vertex $v'$ such that ${v_\lambda}^i \prec v' \prec {v_\lambda}^{i+1}$ for $i \geq 0$
- Label $v_l = {v_\lambda}^m$, where the path $\lambda$ consists of $m$ edges of the form $e({v_\lambda}^i, {v_\lambda}^{i+1})$

**Definition 1.** For any $v_1, v_2 \in S(T)$, $v_1$ and $v_2$ are termed *equivalent*, *iff* there is a bijection $\psi$ which maps the set of paths (and component edges)
$\Lambda_1 = \{\lambda_1(v_1, v_{l_1}) \mid v_{l_1} \in V(T)\backslash S(T)\}$ onto $\Lambda_2 = \{\lambda_2(v_2, v_{l_2}) \mid v_{l_2} \in V(T)\backslash S(T)\}$ in such a way that:
(a) $\psi(e({v_{\lambda_1}}^i, {v_{\lambda_1}}^{i+1})) = e(\psi({v_{\lambda_1}}^i), \psi({v_{\lambda_1}}^{i+1}))$
$= e({v_{\lambda_2}}^i, {v_{\lambda_2}}^{i+1})$ for $0 \leq i \leq m(\lambda)$
(b) $\pi({v_{\lambda_1}}^{i+1} \mid {v_{\lambda_1}}^i) = \pi({v_{\lambda_2}}^{i+1} \mid {v_{\lambda_2}}^i)$
where ${v_{\lambda_1}}^{i+1}$ and ${v_{\lambda_2}}^{i+1}$ label the same **value** on the sample spaces $\mathbb{X}(v^i_{\lambda_1})$ and $\mathbb{X}(v^i_{\lambda_2})$ for $i \geq 0$.

The set of equivalence classes induced by the bijection $\psi$ is denoted $K(T)$, and the elements of $K(T)$ are called *positions*.

**Definition 2.** For any $v_1, v_2 \in S(T)$, $v_1$ and $v_2$ are termed *stage-equivalent*, *iff* there is a bijection $\phi$ which maps the set of edges
$E_1 = \{e_1(v_1, {v_1}') \mid {v_1}' \in \mathbb{X}(v_1)\}$ onto $E_2 = \{e_2(v_2, {v_2}') \mid {v_2}' \in \mathbb{X}(v_2)\}$ in such a way that:
$\pi({v_1}' \mid v_1) = \pi(\phi({v_1}') \mid \phi(v_1)) = \pi({v_2}' \mid v_2)$
where ${v_1}'$ and ${v_2}'$ label the same **value** on the sample spaces $\mathbb{X}(v_1)$ and $\mathbb{X}(v_2)$.

The set of equivalence classes induced by the bijection $\phi$ is denoted $L(T)$, and the elements of $L(T)$ are called *stages*.

A CEG $C(T)$ of our model is constructed as follows:
(1) $V(C(T)) = K(T) \cup \{w_\infty\}$
(2) Each $w, w' \in K(T)$ will correspond to a set of $v, v' \in S(T)$. If, for such $v, v'$, $\exists$ a directed edge $e(v, v') \in E(T)$, then $\exists$ a directed edge $e(w, w') \in E(C(T))$
(3) If $\exists$ an edge $e(v, v_l) \in E(T)$ *st* $v \in S(T)$

and $v_l \in V(T) \backslash S(T)$, then $\exists$ a directed edge $e(w, w_\infty) \in E(C(T))$

(4) If two vertices $v_1, v_2 \in S(T)$ are stage-equivalent, then $\exists$ an undirected edge $e(w_1, w_2) \in E(C(T))$

(5) If $w_1$ and $w_2$ are in the same stage (ie: if $v_1, v_2$ are stage-equivalent in $E(T)$), and if $\pi({v_1}' \mid v_1) = \pi({v_2}' \mid v_2)$ then the edges $e(w_1, {w_1}')$ and $e(w_2, {w_2}')$ have the same *label* or *colour* in $E(C(T))$.

Note that in our example, $w_2$ and $w_3$ are in the same stage and that $\pi(v_6|v_2) = \pi(v_8|v_3)$, $\pi(v_7|v_2) = \pi(v_9|v_3)$, $\pi(v_{18}|v_2) = \pi(v_{23}|v_3)$, so the edges $e(w_2, w_4)$ and $e(w_3, w_4)$ are coloured the same, as are the edges $e(w_2, w_5)$ and $e(w_3, w_6)$ and as are $e(w_2, w_\infty)$ and $e(w_3, w_\infty)$.

More detail on CEG construction can be found in Smith & Anderson (2006), as can a detailed description of how CEGs are read. We conclude section 2 of this paper by looking at two ideas that will be used extensively in the next section.

Firstly, when we say that a position $w$ in our CEG or the set of edges leaving $w$ have an associated *variable*, we are not refering to the measurement-variables of a BN-representation of the problem, each of which must take a value for any atomic event, but to a more flexible construct defined through stage-equivalence in the underlying tree. The exit-edges of a position are simply the collection of possible immediate outcomes in the next step of the process given the history up to that position. A *setting* (or *value* or *level*) is then simply a possible realisation of a variable in this collection.

Secondly, we use these edge probabilities to define the probabilities of composite events in the path sigma field of our CEG:

**Definition 3.** For two positions $w, w'$ with $w \prec w'$, let $\pi_\lambda(w' \mid w)$ be the probability associated with the path $\lambda(w, w')$. Note that this will be a product of edge probabilities.

$$\text{Define} \quad \pi(w' \mid w) \triangleq \sum_{\lambda \in \Lambda} \pi_\lambda(w' \mid w)$$

where $\Lambda$ is the set of all paths from $w$ to $w'$.

Note that the combination rules for path probabilities on CEGs (directly analogous to those for trees) give us that for any 3 positions $w_1, w_2, w_3$, with $w_1 \prec w_2 \prec w_3$, we have that $\pi(w_3 \mid w_1, w_2) = \pi(w_3 \mid w_2)$; that is the probability that we pass through position $w_3$ given that we have passed through positions $w_1$ and $w_2$ is simply the probability that we pass through position $w_3$ given that we have passed through position $w_2$.

## 3 Manipulations of CEGs

The simplest types of intervention are of the form *Do* $X = x_0$ for some variable $X$ and setting $x_0$, and these are really the only interventions that can be satisfactorily analysed using BNs. In this paper we consider a much more general intervention where not only the setting of the manipulated variable, but the variable itself may be different depending on the settings of other variables within the system.

We can model such interventions by the production of a *manipulated* CEG $\hat{C}$ in parallel with our *idle* CEG $C$. In the intervention considered here every path in our CEG is manipulated by having **one** component edge given a probability of 1 or 0. All edges with zero probabilities and branches stemming from such edges are removed (or *pruned*) from $\hat{C}$ (note that in this paper all edges on any CEG will have non-zero probabilities). We will call such an intervention a *singular* manipulation, and denote it *Do Int.*

**Definition 4.** A subset $W_X$ of positions of $C$ qualifies as a singular manipulation set if:

(1) all root-to-sink paths in $C$ pass through exactly one position in $pa(W_X)$, where $w \in pa(W_X)$ if $w \prec w'$ for some $w' \in W_X$ and there exists an edge $e(w, w')$

(2) each position in $pa(W_X)$ has exactly one child in $W_X$, by which we mean that for $w \in pa(W_X)$, there exists exactly one $w' \in W_X$ such that there exists an edge $e(w, w')$

A singular manipulation is then an intervention such that:

(a) for each $w \in pa(W_X)$ and corresponding $w' \in W_X$, $\hat{\pi}(w' \mid w) = 1$

(b) for any $w \in pa(W_X)$ and $w' \notin W_X$ such that $w \prec w'$ and there exists an edge $e(w, w')$, then $\hat{\pi}(w' \mid w) = 0$, and this edge is removed (or *pruned*) in $\hat{C}$

(c) for any $w \notin pa(W_X)$ and $w'$ such that $w \prec w'$ and there exists an edge $e(w, w')$, then $\hat{\pi}(w' \mid w) = \pi(w' \mid w)$

where $\hat{\pi}$ is a probability in our manipulated CEG $\hat{C}$.

Let $W_X = \{w_j\}$, $pa(W_X) = \{w_j^i\}$. Each position in $pa(W_X)$ has exactly one child in $W_X$ so elements of $pa(W_X)$ can be intially identified by their child in $W_X$ (ie by the index $j$). But a position in $W_X$ could have more than one parent in $pa(W_X)$, so we distinguish these parents by a second index $i$. For each pair $(w_j^i, w_j)$ let $X_j^i$ be the variable associated with the edge $e(w_j^i, w_j)$ and $x_j^i$ be the setting of this variable on this edge.

If we also consider a response variable $Y$ downstream from the set of positions $W_X$, then we can show (using for example Pearl's definition of $Do$) that:

$$\pi(y \mid Do\ Int) = \sum_{i,j} \Big[\pi(w_j^i \mid w_0)\ \pi(y \mid w_j)\Big] \quad (3.1)$$

Pearl's own Back Door expression (below) is a simplification of the general manipulated-probability expression used with BNs.

$$\pi(y \mid Do\ x_0) = \sum_z \pi(y \mid z, x_0)\ \pi(z) \quad (3.2)$$

$Z$ here is a subset of the measurement-variables of the BN which obey certain conditions. If $Z$ is chosen carefully then we can calculate $\pi(y \mid Do\ x_0)$ without conditioning on the full set of measurement-variables.

In this paper we use the topology of the CEG to produce an analogous expression to (3.2) for our more general *singular* manipulation, by using a set of positions $W_Z$ downstream from the intervention which can stand-in for the set of positions $W_X$ used in expression (3.1). As with Pearl's expression, the use of such a set $W_Z$ will reduce the complexity of the general expression (3.1) as well as possibly allowing us to sidestep identifiability problems associated with it.

Following Pearl, we have two conditions, which if satisfied, are sufficient for $W_Z$ to be considered a Back Door blocking set. We give the first here, and the second following a few further definitions.

(A) For all $w_j \in W_X$, every $w_j - w_\infty$ path in $C$ must pass through exactly one position $w_k \in W_Z$

The obvious notation for use with CEGs is a path-based one. However most practitioners will be more familiar with expressions such as (3.2), so we here develop a few ideas to allow us to express our causal expression in a similar fashion. The first step in this process is to note that any position $w$ in a CEG has a **unique** set $q(w)$ associated with it, where:

- $Q(w)$ is the **minimum** set of variables, by specifying the settings (or *values* or *levels*) of which, we can describe the union of all $w_0 - w$ paths
- $q(w)$ are the settings of $Q(w)$ which fully describe the union of all $w_0 - w$ paths

Formally this means that:

$$q(w) = \bigcup_{\lambda \in \Lambda_w} q(\lambda)$$

where $q(\lambda)$ are the settings on the $w_0 - w$ path $\lambda$, and $\Lambda_w$ is the set of all $w_0 - w$ paths.

Letting $Z(w)$ be the set of variables encountered on edges upstream of $w$, $X(w)$ be the set of variables encountered on edges downstream of $w$, and $R(w) = Z(w) \backslash Q(w)$, we note that the conditional independence statement encoded by the position $w$ is of the form:

$$X(w) \amalg R(w) \mid q(w)$$

In the CEG in Figure 2 for example, $X(w_4) \amalg R(w_4) \mid q(w_4)$ tells us that product quality is independent of the monitoring system responses, given that B is not replaced.

Note that the definition of $q(w)$ means that the variable-settings within it might not always correspond to simple values of the variables within $Q(w)$. None-the-less, we have

found that $q(w)$ is typically simpler than each individual $q(\lambda)$.

We now use the ideas outlined above to expand the expression (3.1). As the position $w_k$ is uniquely defined by $q(w_k)$, we can write, without ambiguity $\pi(y \mid w_k) = \pi(y \mid q(w_k))$. Using condition (A) we get:

$$\begin{aligned}
&\pi(y \mid Do\ Int) \qquad (3.3)\\
&= \sum_{i,j} \Big[ \pi(w_j^i \mid w_0) \sum_k \pi(w_k, y \mid w_j) \Big]\\
&= \sum_{i,j,k} \pi(w_j^i \mid w_0)\ \pi(y \mid w_j, w_k)\ \pi(w_k \mid w_j)\\
&= \sum_{i,j,k} \pi(w_j^i \mid w_0)\ \pi(y \mid w_k)\ \pi(w_k \mid w_j)\\
&= \sum_k \Big[ \sum_{i,j} \pi(w_j^i|w_0)\pi(w_k|w_j) \Big] \pi(y|q(w_k))
\end{aligned}$$

The equivalence of $\pi(y \mid w_j, w_k)$ and $\pi(y \mid w_k)$ is a consequence of the equivalence of $\pi(w_3 \mid w_1, w_2)$ and $\pi(w_3 \mid w_2)$ noted at the end of section 2, and proved in Thwaites & Smith (2006b).

We now need a number of technical definitions before we can introduce our second condition and proceed to our causal expression. Examples illustrating these definitions can be found in Thwaites & Smith (2006b).

Now, the position $w_k$ can also be fully described by the union of disjoint events, each of which is (by conditions (1) and (A)) a $w_0 - w_j^i - w_k$ path for some $w_j^i \in pa(W_X)$. These events divide into 2 distinct sets:

(1) $w_0 - w_j^i - w_j - w_k$ paths

(2) paths that do **not** utilise the $x_j^i$ edge when leaving $w_j^i$ (formally $w_0 - w_j^i - w' - w_k$ paths where there exists an edge $e(w_j^i, w')$, but $w' \notin W_X$).

We can combine the events in set (1) into *composite* or *C-paths* so that each *C-path* passes through exactly one $w_j^i$ and can be uniquely characterised by a pair $q_j^i(w_k) = (x_j^i, z_j^i(w_k))$, where $z_j^i$ is defined as follows:

- $Z_j^i(w_k)$ is the **minimum** set of variables, by specifying the settings of which, we can describe the union of all $w_0 - w_j^i - w_j - w_k$ paths (with $X_j^i$ excluded from this set)
- $z_j^i(w_k)$ are the settings of $Z_j^i(w_k)$ which (with the addition of $X_j^i = x_j^i$) fully describe the union of all $w_0 - w_j^i - w_j - w_k$ paths

**Definition 5.** We express this formally as:
Let $q_j^i(w_k) = \bigcup_{\lambda \in \Lambda} q(\lambda)$, where $q(\lambda)$ are the settings on the $w_0 - w_j^i - w_j - w_k$ path $\lambda$, and $\Lambda$ is the set of all $w_0 - w_j^i - w_j - w_k$ paths in C. Let $Q_j^i(w_k)$ be the set of variables present in $q_j^i(w_k)$.
Define $Z_j^i(w_k)$ as $Q_j^i(w_k) \backslash X_j^i$. Let $z_j^i(w_k)$ be the settings of $Z_j^i(w_k)$ compatible with $q_j^i(w_k)$.

Our $X_j^i, x_j^i, Z_j^i(w_k), z_j^i(w_k)$ are directly analogous to Pearl's $X, x, Z$ and $z$ in expression (3.2), and fulfil similar roles in our final causal expression.

The following rather technical definitions are **only** required for an understanding of the proof. Definition 7 deals with the idea of a *descendant* which is very similar to the analogous idea in BNs, and is needed for condition (B).

**Definition 6.**
Define $q(w_j^i)$ analogously with the definition of $q(w)$. Let $Q(w_j^i)$ be the set of variables present in $q(w_j^i)$.
Define $Q_j(w_k)$ as $Z_j^i(w_k) \backslash Q(w_j^i)$. Note that $Q(w_j^i) \subset Z_j^i(w_k)$. Let $q_j(w_k)$ be the settings of $Q_j(w_k)$ compatible with $z_j^i(w_k)$ (or $q_j^i(w_k)$).

We can therefore write:

$$z_j^i(w_k) = (q(w_j^i), q_j(w_k))$$

$$q_j^i(w_k) = (x_j^i, z_j^i(w_k)) = (q(w_j^i), x_j^i, q_j(w_k))$$

We can also combine the events in set (2) into *C-paths*, each of which can be uniquely characterised by $r_j^i(w_k) = \bigcup_{\mu \in M} q(\mu)$, where $q(\mu)$ are the settings on the $w_0 - w_j^i - w' - w_k$ path $\mu$, and $M$ is the set of all $w_0 - w_j^i - w' - w_k$ paths in C. We can therefore write:

$$q(w_k) = \Big[ \bigcup_{i,j} q_j^i(w_k) \Big] \bigcup \Big[ \bigcup_{i,j} r_j^i(w_k) \Big]$$

**Definition 7.** Consider variables $A, D, \{B_m\}$ defined on our CEG $C$. Then $D$ is a descendant of $A$ in $C$ if there exists a sequence

of (not necessarily adjacent) edges $e_1, \ldots e_n$ forming part of a $w_0 - w_\infty$ path in $C$ where the edges $e_1, \ldots e_n$ are labelled respectively $b_1 \mid (a, \ldots)$, $b_2 \mid (b_1, \ldots), \ldots b_{n-1} \mid (b_{n-2}, \ldots)$, $d \mid (b_{n-1}, \ldots)$, or if there exists an edge forming part of a $w_0 - w_\infty$ path in $C$ labelled $d \mid (a, \ldots)$; where $a, b_1, b_2, \ldots b_{n-1}, d$ are settings of $A, B_1, B_2, \ldots B_{n-1}, D$.

We are now in a position to state our 2nd condition.

(B) In the sub-CEG $C_j^i$ with $w_j^i$ as root-node, $Q_j(w_k)$ must contain no descendants of $X_j^i$ for all $i, j$, for each position $w_k$

Checking that condition (B) is fulfilled is actually straightforward on a CEG, especially since we will know which manipulations we intend to investigate, and can usually construct our CEG so as to make $Q_j(w_k)$ as *small* as possible for all values of $j, k$.

We can now replace expression (3.3) by a Back Door expression for singular manipulations:

**Proposition 1.**

$$\begin{aligned} &\pi(y \mid Do\ Int) \qquad (3.4)\\ &= \sum_k \Big[\sum_{i,j} \pi(z_j^i(w_k))\Big]\ \pi(y \mid q(w_k)) \end{aligned}$$

*Proof.*
Consider
$\pi(w_k \mid w_j) = \pi(w_k \mid w_j^i, w_j)$
$= \pi(q(w_k) \mid q(w_j^i), x_j^i)$
$= \pi\Big(\Big[\bigcup_{m,n} q_n^m(w_k)\Big] \cup \Big[\bigcup_{m,n} r_n^m(w_k)\Big] | q(w_j^i), x_j^i\Big)$
$= \sum_{m,n} \pi(q_n^m(w_k) \mid q(w_j^i), x_j^i)$
$\quad + \sum_{m,n} \pi(r_n^m(w_k) \mid q(w_j^i), x_j^i)$
since disjoint.
$= \pi(q_j^i(w_k) | q(w_j^i), x_j^i) + \pi(r_j^i(w_k) | q(w_j^i), x_j^i)$
$= \pi(q_j^i(w_k) \mid q(w_j^i), x_j^i)$
$= \pi(q(w_j^i), x_j^i, q_j(w_k) \mid q(w_j^i), x_j^i)$
$= \pi(q_j(w_k) \mid q(w_j^i), x_j^i)$

But this is simply the probability that $Q_j(w_k) = q_j(w_k)$ given that $X_j^i = x_j^i$ in the sub-CEG $C_j^i$.
Condition (B) implies that $X_j^i \amalg Q_j(w_k)$ in $C_j^i$ since $X_j^i$ has no parents in this CEG. So we get:

$$\pi(w_k \mid w_j) = \pi(q_j(w_k) \mid q(w_j^i))$$

Substituting this into expression (3.3), we get:

$$\begin{aligned} &\pi(y \mid Do\ Int)\\ &= \sum_k \Big[\sum_{i,j} \pi(w_j^i \mid w_0)\ \pi(q_j(w_k) \mid q(w_j^i))\Big]\\ &\qquad \times \pi(y \mid q(w_k))\\ &= \sum_k \Big[\sum_{i,j} \pi(q(w_j^i))\ \pi(q_j(w_k) \mid q(w_j^i))\Big]\\ &\qquad \times \pi(y \mid q(w_k))\\ &= \sum_k \Big[\sum_{i,j} \pi(z_j^i(w_k))\Big]\ \pi(y \mid q(w_k)) \qquad \square \end{aligned}$$

It is possible to show that the expression $\pi(y \mid q(w_k))$ can be replaced by a probability conditioned on a **single** $w_0 - w_k$ path, and moreover that even on that path $Y$ may well be independent of some of the variables encountered given the path-settings of the others — for details see Thwaites & Smith (2006b).

We also noted earlier that $q(w) = \bigcup q(\lambda)$ is typically simpler than each individual $q(\lambda)$. In most instances $\sum_{i,j} \pi(z_j^i(w_k))$ will be the probability of a union of disjoint events which will also typically be simpler than an individual $z_j^i(w_k)$. We can deduce that calculating $\sum_{i,j} \pi(z_j^i(w_k))$ is unlikely to be a complex task.

We conclude this section by demonstrating how expression (3.4) is related to Pearl's Back Door expression (3.2):
Consider the intervention $Do\ X = x_0$, and let $X_j^i = X$ and $x_j^i = x_0$ for all $i, j$. Combine all our $w_0 - w_j^i - w_j - w_k$ *C-paths*, and write $\bigcup_{i,j} q_j^i(w_k) = (x_0, z(w_k))$. Rephrase conditions (2) and (B) as:

(2) each position in $pa(W_X)$ has exactly one of its outward edges in $C$ labelled $x_0$, and this edge joins the position in $pa(W_X)$ to a position in $W_X$

(B) $Z(w_k)$ must contain no descendants of $X$ (where $Z(W_k)$ is defined from $z(w_k)$ in the obvious manner)

Then with a little work, we can replace expression (3.4) by:

$$\pi(y \mid Do\ Int) = \sum_k \pi(z(w_k))\ \pi(y \mid x_0, z(w_k))$$

If $Z(w_k)$ contains the same variables for all $k$, and $z(w_k)$ runs through all settings of $Z(w_k)$ as we run through all $w_k$, then this expression reduces to Pearl's expression (3.2).

## 4 Causal Analysis on CEGs and BNs

The principal advantage that CEGs have over BNs when it comes to Causal analysis is their flexibility. In a BN the kind of manipulations we can consider are severely restricted (see for example section 2.6 of Lauritzen (2001) where he comments on Shafer (1996)), whereas using a CEG we can tackle not only the conventional $Do\ X = x_0$ manipulations of symmetric models, but also the analysis of interventions on asymmetric models and manipulations where both the manipulated-variable and the manipulated-variable value can differ for different settings of other variables. It is also the case that our blocking sets are sets of values or settings, and do not need to correspond to any fixed subset of the original problem random variables.

For simplicity of exposition in this paper we have not fully exploited the potential flexibility of the CEG, considering only formulae associated with a blocking-set $W_Z$ of *positions* downstream of $W_X$. We can also consider sets of *stages* upstream of $W_X$, and combinations of the two. Also, we have only discussed one particular fairly coarse example of an intervention. There are often circumstances where some paths in our CEG are not manipulated at all, for example in a treatment regime where only patients with certain combinations of symptoms (ie at certain positions or stages) are treated. There are also non-singular interventions where (for instance) a manipulation, rather than forcing a path to follow one specific edge at some vertex, instead provides a probability distribution for the outgoing edges of that vertex.

So not only are CEGs ideal representations of asymmetric discrete models, retaining the convenience of a tree-form for describing how a process unfolds but also expressing a rich collection of the model's conditional independence properties, but their event-based causal analysis has distinct advantages over the variable-based analysis one performs on Bayesian Networks.

## References


A.P. Dawid. 2002. Influence Diagrams for Causal Modelling and Inference. *International Statistical Review 70.*

M. Jaeger. 2002. Probabilistic Decision Graphs — Combining Verification and AI techniques for Probabilistic Inference. In *PGM '02 Proceedings of the 1st European Workshop on Probabilistic Graphical Models,* pages 81-88.

S.L. Lauritzen. 2001. Causal Inference from Graphical Models. In O.E. Barndorff-Nielsen et al (Eds) *Complex Stochastic Systems.* Chapman and Hall / CRC.

J. Pearl. 2000. *Causality: models, reasoning and inference.* Cambridge University Press.

E. Riccomagno and J.Q. Smith. 2005. The Causal Manipulation and Bayesian Estimation of Chain Event Graphs. CRiSM Research Report no. 05-16, University of Warwick.

G. Shafer. 1996. *The Art of Causal Conjecture.* MIT Press.

J.Q. Smith and P.E. Anderson. 2006. Conditional independence and Chain Event Graphs. Accepted, subject to revision, by *Journal of Artificial Intelligence.*

P.A. Thwaites and J.Q. Smith. 2006a. Non-symmetric models, Chain Event Graphs and propagation. In *IPMU '06 Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems.*

P.A. Thwaites and J.Q. Smith. 2006b. Singular manipulations on Chain Event Graphs. Warwick Research Report, University of Warwick.

# Severity of Local Maxima for the EM Algorithm: Experiences with Hierarchical Latent Class Models

Yi Wang and Nevin L. Zhang
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
{wangyi,lzhang}@cse.ust.hk

**Abstract**

It is common knowledge that the EM algorithm can be trapped at local maxima and consequently fails to reach global maxima. We empirically investigate the severity of this problem in the context of hierarchical latent class (HLC) models. Our experiments were run on HLC models where dependency between neighboring variables is strong. (The reason for focusing on this class of models will be made clear in the main text.) We first ran EM from randomly generated single starting points, and observed that (1) the probability of hitting global maxima is generally high, (2) it increases with the strength of dependency and sample sizes, and (3) it decreases with the amount of extreme probability values. We also observed that, at high dependence strength levels, local maxima are far apart from global ones in terms of likelihoods. Those imply that local maxima can be reliably avoided by running EM from a few starting points and hence are not a serious issue. This is confirmed by our second set of experiments.

## 1 Introduction

The EM algorithm (Dempster et al., 1977) is a popular method for approximating maximum likelihood estimate in the case of incomplete data. It is widely used for parameter learning in models, such as mixture models (Everitt and Hand, 1981) and latent class models (Lazarsfeld and Henry, 1968), that contain latent variables.

A well known problem associated with EM is that it can be trapped at local maxima and consequently fails to reach global maxima (Wu, 1983). One simple way to alleviate the problem is to run EM many times from randomly generated starting points, and take the highest likelihood obtained as the global maximum. One problem with this method is that it is computationally expensive when the number of starting points is large because EM converges slowly. For this reason, researchers usually adopt the *multiple restart* strategy (Chickering and Heckerman, 1997; van de Pol et al., 1998; Uebersax, 2000; Vermunt and Magidson, 2000): First run EM from multiple random starting points for a number of steps, then pick the one with the highest likelihood, and continue EM from the picked point until convergence. In addition to multiple restart EM, several more sophisticated strategies for avoiding local maxima have also been proposed (Fayyad et al., 1998; Ueda and Nakano, 1998; Ueda et al., 2000; Elidan et al., 2002; Karciauskas et al., 2004).

While there is abundant work on avoiding local maxima, we are aware of few work on the severity of the local maxima issue. In this paper, we empirically investigate the severity of local maxima for hierarchical latent class (HLC) models. Our experiments were run on HLC models where dependency between neighboring variables is strong. This class of models was chosen because we use HLC models to discover latent structures. It is our philosophical view that one cannot expect to discover latent structures reliably unless observed variables strongly depend on latent variables (Zhang, 2004; Zhang and Kocka, 2004).

In the first set of experiments, we ran EM from randomly generated single starting points, and observed that (1) the probability of hitting global maxima is generally high, (2) it increases with the strength of dependency and sample sizes, and (3) it decreases with the amount of extreme probability values. We also observed that, at high dependence strength levels, local maxima are far apart from global ones in terms of likelihoods.

Those observations have immediate practical implications. Earlier in this section, we mentioned a simple local-maxima avoidance method. We pointed out one of its problems, i.e. its high computational complexity, and said that multiple restart can alleviate this problem. There is another problem with the method: there is no guidance on how many starting points to use in order to avoid local maxima reliably. Multiple restart provides no solution for this problem.

Observations from our experiments suggest a guideline for strong dependence HLC models. As a matter of fact, they imply that local maxima can be reliably avoided by using multiple restart and a few starting points. This is confirmed by our second set of experiments.

The remainder of this paper is organized as follows. In Section 2, we review some basic concepts about HLC models and the EM algorithm. In Sections 3 and 4, we report our first and second sets of experiments respectively. We conclude this paper and point out some potential future work in Section 5.

## 2 Background

### 2.1 Hierarchical Latent Class Models

*Hierarchical latent class (HLC) models* (Zhang, 2004) are tree-structured Bayesian networks where the leaf nodes are observed while the internal nodes are hidden. An example HLC model is shown in Figure 1. Following the conventions in latent variable model literatures, we call the leaf nodes *manifest variables* and the internal nodes *latent variables*[1].

[1]In this paper, we do not distinguish between nodes and variables.

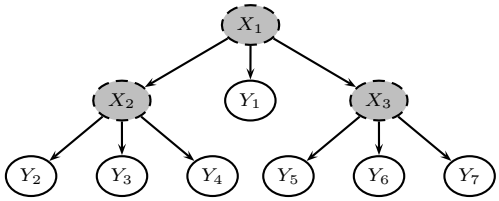



Figure 1: An example HLC model. $X_1$, $X_2$, $X_3$ are latent variables, and $Y_1$, $Y_2$, $\cdots$, $Y_7$ are manifest variables.

We usually write an HLC model as a pair $M=(m, \boldsymbol{\theta})$. The first component $m$ consists of the model structure and cardinalities of the variables. The second component $\boldsymbol{\theta}$ is the collection of parameters. Two HLC models $M=(m, \boldsymbol{\theta})$ and $M'=(m', \boldsymbol{\theta}')$ are *marginally equivalent* if they share the same set of manifest variables $\mathbf{Y}$ and $P(\mathbf{Y}|m, \boldsymbol{\theta})=P(\mathbf{Y}|m', \boldsymbol{\theta}')$.

We denote the cardinality of a variable $X$ by $|X|$. For a latent variable $X$ in an HLC model, denote the set of its neighbors by $\mathbf{nb}(X)$. An HLC model is *regular* if for any latent variable $X$, $|X| \leq \frac{\prod_{Z \in \mathbf{nb}(X)} |Z|}{\max_{Z \in \mathbf{nb}(X)} |Z|}$, and the inequality strictly holds when $X$ has only two neighbors, one of which being a latent variable. Zhang (2004) has shown that an irregular HLC model can always be reduced to a marginally equivalent HLC model that is regular and contains fewer independent parameters. Henceforth, our discussions are restricted to regular HLC models.

### 2.2 Strong Dependence HLC Models

The strength of dependency between two variables is usually measured by mutual information or correlation (Cover and Thomas, 1991). However, there is no general definition of strong dependency for HLC models yet. In this study, we use the operational definition described in the following paragraph.

Consider a probability distribution. We call the component with the largest probability mass the *major component*. We say that an HLC model is a *strong dependence model* if

- The cardinality of each node is no smaller than that of its parent.
- The major components in all conditional distributions are larger than 0.5, and

- In each conditional probability table, the major components of different rows are located in different columns.

In general, the larger the major components, the higher the dependence strength (DS) level. In the extreme case, when all major components are equal to 1, the HLC model becomes deterministic. Strong dependence HLC models defined in this way have been examined in our previous work on discovering latent structures (Zhang, 2004; Zhang and Kocka, 2004). The results show that such models can be reliably recovered from data.

### 2.3 EM Algorithm

Latent variables can never be observed and their values are always missing in data. This fact complicates the maximum likelihood estimation problem, since we cannot compute sufficient statistics from incomplete data records. A common method to deal with such situations is to use the *expectation-maximization (EM) algorithm* (Dempster et al., 1977). The EM algorithm starts with a randomly chosen estimation to parameters, and iteratively improves this estimation by increasing its loglikelihood. In each EM step, the task of increasing loglikelihood is delegated to the maximization of the *expected loglikelihood function*. The latter is a lower bound of the loglikelihood function. It is defined as

$$\begin{aligned} & Q(\boldsymbol{\theta}|\mathcal{D},\boldsymbol{\theta}^t) \\ = & \sum_{l=1}^{m}\sum_{\mathbf{X}_l} P(\mathbf{X}_l|\mathbf{D}_l,\boldsymbol{\theta}^t)\log P(\mathbf{X}_l,\mathbf{D}_l|\boldsymbol{\theta}), \end{aligned}$$

where $\mathcal{D}=\{\mathbf{D}_1,\mathbf{D}_2,\cdots,\mathbf{D}_m\}$ denotes the colletion of data, $\mathbf{X}_l$ denotes the set of variables whose values are missing in $\mathbf{D}_l$, and $\boldsymbol{\theta}^t$ denotes the estimation at the $t$-th step. The EM algorithm terminates when the increase in loglikelihood between two successive steps is smaller than a predefined stopping threshold.

It is common knowledge that the EM algorithm can be trapped at local maxima and consequently fails to reach global maxima (Wu, 1983). The specific result depends on the choice of the starting point. A common method to avoid local maxima is to run EM many times with randomly generated starting points, and pick the instance with the highest likelihood as the final result. The more starting points it uses, the higher the chance it can reach the global maximum. However, due to the slow convergence of EM, this method is computationally expensive. A more feasible method, called *multiple restart EM*, is to run EM with multiple random starting points, and retain the one with the highest likelihood after a specified number of initial steps. This method and its variant are commonly used to learn latent variable models in practice (Chickering and Heckerman, 1997; van de Pol et al., 1998; Uebersax, 2000; Vermunt and Magidson, 2000). Other work on escaping from poor local maxima includes (Fayyad et al., 1998; Ueda and Nakano, 1998; Ueda et al., 2000; Elidan et al., 2002; Karciauskas et al., 2004).

## 3 Severity of Local Maxima

Here is the strategy that we adopt to empirically investigate the severity of local maxima in strong dependence HLC models: (1) create a set of strong dependence models, (2) sample some data from each of the models, (3) learn model parameters from the data by running EM to convergence from a number of starting points, (4) graph the final loglikelihoods obtained.

The final loglikelihoods for different starting points could be different due to local maxima. Hence, an inspection of their distribution would give us a good idea about the severity of local maxima.

### 3.1 Experiment Setup

The structure of all models used in our experiments was the ternary tree with height equals 3, as shown in Figure 2. The cardinalities of all variables were set at 3. Parameters were randomly generated subject to the strong dependency condition. We examined 5 DS levels, labeled from 1 to 5. They correspond to restricting the major components within the following 5 intervals: $[0.5, 0.6)$, $[0.6, 0.7)$,$[0.7, 0.8)$,$[0.8, 0.9)$, and $[0.9, 1.0]$. For each DS level, 5 different parameterizations were generated. Consequently,

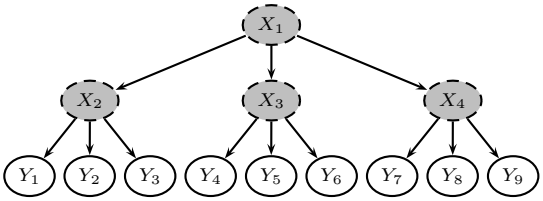



Figure 2: The structure of generative models.

we examined 25 models in total.

From each of the 25 models, four training sets of 100, 500, 1000, and 5000 records were sampled. On each training set, we ran EM independently for 100 times, each time starting from a randomly selected initial point. The stopping threshold of EM was set at 0.0001. The highest loglikelihood obtained in the 100 runs is regarded as the global maximum.

### 3.2 Results

The results are summarized in Figures 3 and 4. In Figure 3, there is a plot for each combination of DS level (row) and sample size (column). As mentioned earlier, 5 models were created for each DS level. The plot is for one of those 5 models. In the plot, there are three curves: solid, dashed, and dotted[2]. The dashed and dotted curves are for Section 4. The solid curve is for this section. The curve depicts a distribution function $F(x)$, where $x$ is loglikelihood and $F(x)$ is the percent of EM runs that achieved a loglikelihood no larger than $x$.

While a plot in Figure 3 is about one model at a DS level, a plot in Figure 4 represents an aggregation of results about all 5 models at a DS level. Loglikelihoods for different models can be in very different ranges. To aggregate them in a meaningful way, we introduce the concept of relative likelihood shortfall of EM run.

Consider a particular model. We have run EM 100 times and hence obtained 100 loglikelihoods. The maximum is regarded as the optimum and is denoted by $l^*$. Suppose a particular EM run resulted in loglikelihood $l$. Then the *relative likelihood shortfall* of that EM run is defined as $(l-l^*)/l^*$. This value is nonnegative. The smaller it is, the higher the quality of the parameters produced by the EM run. In particular, the relative likelihood shortfall of the run that produced the global maximum $l^*$ is 0.

For a given DS level, there are 5 models and hence 500 EM runs. We put the relative likelihood shortfalls of all those EM runs into one set and let, for any nonnegative real number $x$, $F(x)$ be the percent of the elements in the set that is no larger than $x$. We call $F(x)$ the *distribution function of (aggregated) relative likelihood shortfalls of EM runs*, or simply *distribution of EM relative likelihood shortfall*, for the DS level.

The first 5 plots in Figure 4 depict the distributions of EM relative likelihood shortfall for the 5 DS levels. There are four curves in each plot, each corresponding to a sample size[3].

#### 3.2.1 Probability of Hitting Global Maxima

The most interesting question is how often EM hits global maxima. To answer this question, we first look at the solid curves in Figure 3. Most of them are stair-shaped. In each curve, the $x$-position of the right most stair is the global maximum, and the height of that stair is the frequency of hitting the global maximum.

We see that the frequency of hitting global maxima was generally high for high DS levels. In particular, for DS level 3 or above, EM reached global maxima more than half of the time. For sample size 500 or above, the frequency was even greater than 0.7. On the other hand, the frequency was low for DS level 1, especially when the sample size was small.

The first 5 plots in Figure 4 tell the same story. In those plots, the global maxima are represented by $x$=0. The heights of the curves at $x$=0 are the frequencies of hitting global maxima. We see that for DS level 3 or above, the frequency of hitting global maxima is larger than 0.5, except that for DS level 3 and sample size 100. And once again, the frequency was low for DS level 1.

[2]Note that in some plot (e.g., that for DS level 4 and sample size 5000) the curves overlap and are indistinguishable.

[3]Note that in Figure 4 (b) and (c) some curves are close to the $y$-axes and are hardly distinguishable.

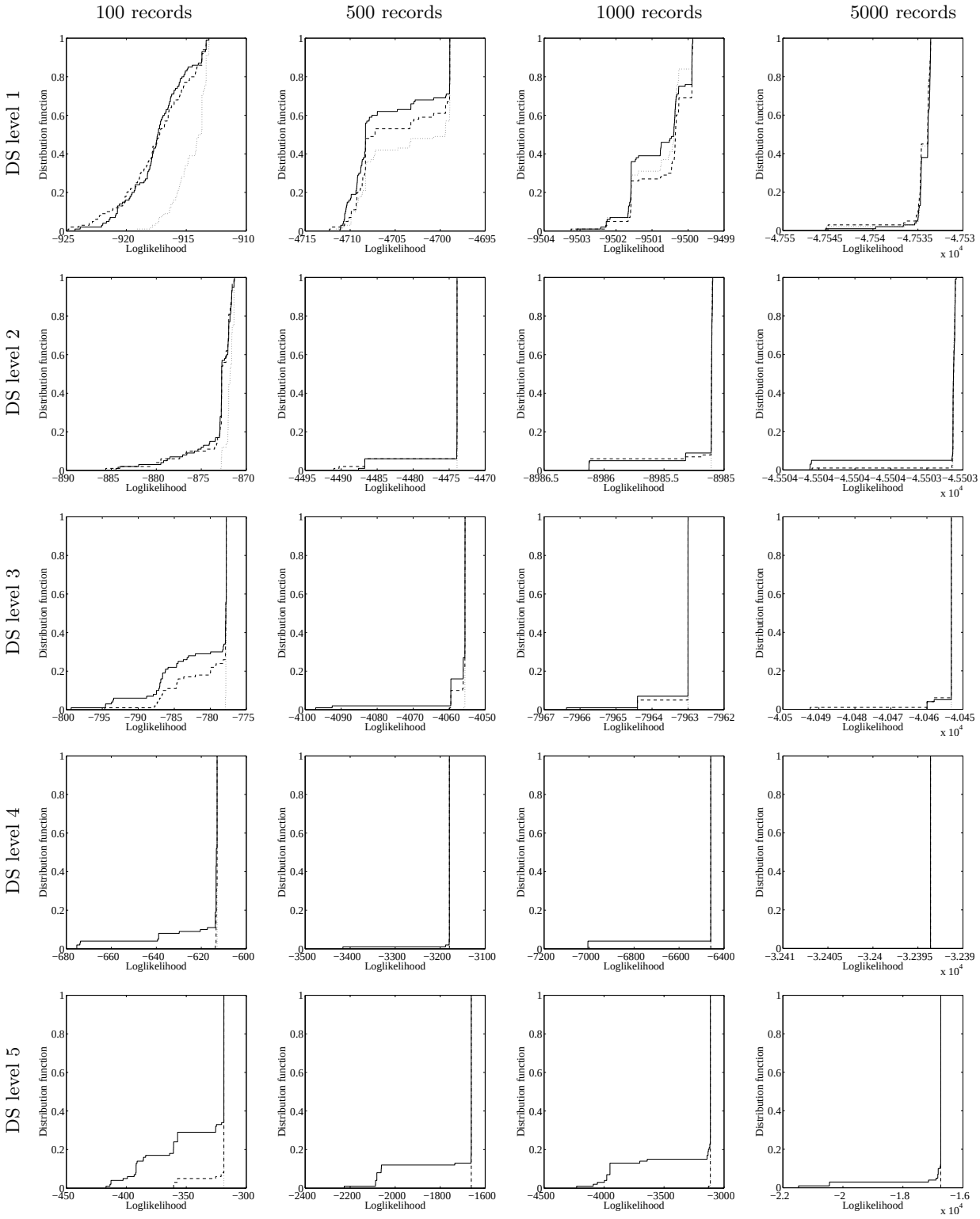
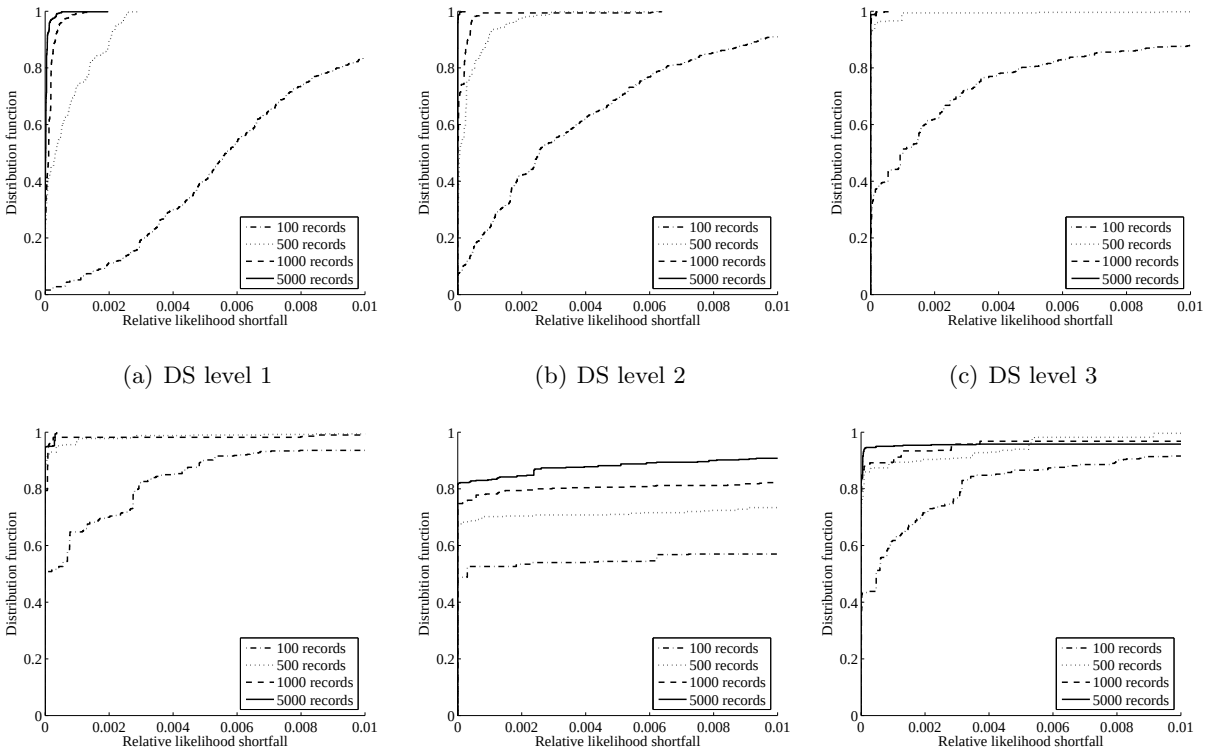



Figure 3: Distributions of loglikelihoods obtained by different EM runs. Solid curves are for EM runs with single starting points. Dashed and dotted curves are for runs of multiple restart EM with setting 4×10 and 16×50 respectively (see Section 4). Each curve depicts a distribution function $F(x)$, where $x$ is loglikelihood and $F(x)$ is the percent of runs that achieved a loglikelihood no larger than $x$.




(a) DS level 1 (b) DS level 2 (c) DS level 3

(d) DS level 4 (e) DS level 5 (f) DS level 3 with 15% zero components

Figure 4: Distributions of EM relative likelihood shortfall for different DS levels.

### 3.2.2 DS Level and Local Maxima

To see how DS level influences the severity of local maxima, read each column of Figure 3 (for a given sample size) from top to bottom. Notice that, in general, the height of the right most stair increases with the DS level. It means that the frequency of hitting global maxima increases with DS level. The same conclusion can be read off from Figure 4. Compare the curves for a given sample size from the first 5 plots. In general, their values at $x$=0 rise up as we move from DS level 1 to DS level 5.

### 3.2.3 Extreme Parameter Values and Local Maxima

There are some exceptions to the regularities mentioned in the previous paragraph. We see in both figures that the frequency of hitting global maxima decreases as the DS level changes from 4 to 5. By comparing Figure 4 (c) and (d), we also find that the frequency slightly drops when the DS level changes from 3 to 4, given that the sample size is large.

We conjecture that this is because that, in generative models for DS levels 4 and 5, there are many parameter values close to 0. It has been reported in latent variable model literatures that extreme parameter values cause local maxima (Uebersax, 2000; McCutcheon, 2002).

To confirm our conjecture, we created another set of five models for DS level 3. The parameters were generated in the same way as before, except that we randomly set 15% of the non-major components in the probability distributions to 0. We then repeated the experiments for this set of models. The EM relative likelihood shortfall distributions are given in Figure 4 (f). We see that, as expected, the frequency of hitting global maxima did decrease considerably compared with Figure 4 (c).

### 3.2.4 Sample Size and Local Maxima

We also noticed the power of the sample size. By going through each row of Figure 3, we found

that the frequency of hitting global maxima increases with the sample size. The phenomenon is more apparent if we look at Figure 4, where curves for different sample sizes are plotted in the same picture. As the sample size goes up, the curve becomes steeper and its value at $x$=0 increases significantly.

#### 3.2.5 Quality of Local Maxima

In addition to the frequency of hitting global maxima, another interesting question is how bad local maxima could be. Let us first examine the local maxima in Figure 3. They are marked by the $x$-positions of the inflection points on curves. For DS level 1, the local maxima are not far from the global ones. The discrepancies are less than 15. Moreover, there are a lot inflection points on the curves, namely, distinct local maximal solutions. They distributed evenly within the interval between the worst local maxima and the global ones.

For the other DS levels, things are different. We first observed that the quality of local maxima can be extremely poor in some situation. The worst case is that of DS level 5 with sample size 5000. The loglikelihood of the local maximum can be lower than that of the global one by more than 4000. The second thing we observed is that the curves contain much fewer inflection points. In other words, there are only a few distinct local maximal solutions in such cases. Moreover, those local maxima stay far apart from global ones since the steps of the staircases are fairly large. Those observations can be confirmed by studying the first 5 plots in Figure 4, where the curves and the inflection points can be interpreted similarly.

## 4 Performance of Multiple Restart EM

We emphasize two observations mentioned in the previous section: (1) the probability of hitting global maxima is generally high, and (2) likelihoods of local maxima are far apart from those of global maxima at high DS levels. We will see that these observations have immediate implications on the performance of multiple restart EM.

We say that a starting point is *optimal* if it converges to the global maximum. The first observation can be restated as follows: the probability for a randomly generated starting point to be optimal is generally high. Consequently, it is almost sure that there is an optimal starting point within a few randomly generated ones.

As it is well known, the EM algorithm increases the likelihood quickly in its early stage and slows down when it is converging. In other words, the likelihood should become relatively stable after a few steps. Therefore, the second observation implies that we can easily separate the optimal starting point from the others after running a few EM steps on them.

A straightforward consequence of the above inference is that multiple restart EM with a few starting points and initial steps should reliably avoid local maxima for strong dependence HLC models. To confirm this conjecture, we ran multiple restart EM independently for 100 times on each training set that was presented in Figure 3. We tested two settings for multiple restart EM: (1) 4 random starting points with 10 initial steps (in short, 4×10), and (2) 16 random starting points with 50 initial steps (in short, 16×50). As before, we plotted the distributions of loglikelihoods in Figure 3. The dashed and the dotted curves denote the results for settings of 4×10 and 16×50, respectively.

From Figure 3, we see that multiple restart EM with setting 16×50 can reliably avoid local maxima for DS level 2 or above. Actually, the dotted curves are parallel to the $y$-axes except that for DS level 2 and sample size 100. It means that global maxima can always be reached in such cases. For setting 4×10, similar behaviors are observed for DS level 4 or 5 and sample size 500 or above. Note that dashed and dotted curves overlap in those plots.

Nonetheless, we also notice that, for DS level 1, multiple restart EM with both settings still can not find global maxima reliably. This is consistent with our reasoning. As we have mentioned in Section 3.2.1, when we ran EM with randomly generated single starting points, the frequency of hitting global maxima is low for DS level 1. In other words, it is hard to hit an op-

timal starting point by chance. Moreover, due to the small discrepancy among local maxima (see Section 3.2.5), it demands a large number of initial steps to distinguish an optimal starting point from the others. Therefore, the effectiveness of multiple restart EM degenerates. In such situations, one can either increase the number of starting points and initial steps, or appeal to more sophisticated methods for avoiding local maxima.

## 5 Conclusions

We have empirically investigated the severity of local maxima for EM in the context of strong dependence HLC models. We have observed that (1) the probability of hitting global maxima is generally high, (2) it increases with the strength of dependency and sample sizes, (3) it decreases with the amount of extreme probability values, and (4) likelihoods of local maxima are far apart from those of global maxima at high dependence strength levels. We have also empirically shown that the local maxima can be reliably avoided by using multiple restart EM with a few starting points and hence are not a serious issue.

Our discussion has been restricted to a specific class of HLC models. In particular, we have defined the strong dependency in an operational way. One can devise more formal definitions and carry on similar studies for generalized strong dependence models. Another future work would be the theoretical exploration to support our experiences.

Based on our observations, we have analyzed the performance of multiple restart EM. One can exploit those observations to analyze more sophisticated strategies for avoiding local maxima. We believe that those observations can also give some inspirations to develop new methods on this direction.

### Acknowledgments

Research on this work was supported by Hong Kong Grants Council Grant #622105.

## References

D. M. Chickering and D. Heckerman. 1997. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29:181–212.

T. M. Cover and J. A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons, Inc.

A. P. Dempster, N. M. Laird, and D. R. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. 2002. Data perturbation for escaping local maxima in learning. In *AAAI-02*, pages 132–139.

B. S. Everitt and D. J. Hand. 1981. *Finite Mixture Distributions*. Chapman and Hall.

U. M. Fayyad, C. A. Reina, and P. S. Bradley. 1998. Initialization of iterative refinement clustering algorithms. In *KDD-98*, pages 194–198.

G. Karciauskas, T. Kocka, F. V. Jensen, P. Larranaga, and J. A. Lozano. 2004. Learning of latent class models by splitting and merging components. In *PGM-04*.

P. F. Lazarsfeld and N. W. Henry. 1968. *Latent Structure Analysis*. Houghton Mifflin.

A. L. McCutcheon. 2002. Basic concepts and procedures in single- and multiple-group latent class analysis. In *Applied Latent Class Analysis*, chapter 2, pages 56–85. Cambridge University Press.

J. S. Uebersax. 2000. A brief study of local maximum solutions in latent class analysis. http://ourworld.compuserve.com/homepages/jsuebersax/local.htm.

N. Ueda and R. Nakano. 1998. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282.

N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. 2000. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128.

F. van de Pol, R. Langeheine, and W. de Jong, 1998. *PANMARK user manual, version 3*. Netherlands Central Bureau of Statistics.

J. K. Vermunt and J. Magidson, 2000. *Latent GOLD User's Guide*. Statistical Innovations Inc.

C. F. J. Wu. 1983. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103.

N. L. Zhang and T. Kocka. 2004. Efficient learning of hierarchical latent class models. In *ICTAI-04*, pages 585–593.

N. L. Zhang. 2004. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5(6):697–723.

# Optimal Design with Design Networks

Y. Xiang
University of Guelph, Canada

**Abstract**

As part of decision-theoretic collaborative design, this work addresses representation and computation issues for a set-based design agent. We refine the definition of design networks to be more expressive for design knowledge and more effective for guiding model construction and verification. We propose a compiled structure of design networks and a suite of algorithms that computes optimal designs efficiently. The result provides a mechanism that combines probabilistic, constraint-based, and decision-theoretic reasoning for single-agent optimal design, and fills in a gap in optimal collaborative design.

## 1 Introduction

This work concerns optimal decision in centralized and collaborative design. At the centralized front, it deals with set-based design (Ward, 1989; Sobek et al., 1999) which, unlike point-based design often confined to local optimals, evaluates all alternative designs to seek the global optimal. Primary challenges for product lifecycle management include how to efficiently propagate constraints and search in a design space (Paredis et al., 2006). This work contributes to meeting these challenges with an algorithm suite that designs optimally decision-theoretically and efficiently.

At the collaborative front, it solves a subproblem in decision-theoretic design: Most research on collaborative design, e.g. (Konduri and Chandrakasan, 1999), focuses on designer information sharing but not on making design choices. Exceptions, such as *Collaborative optimization* (Braun et al., 1996) are essentially point-based and only produce locally optimal designs. A decision-theoretic graphical model, termed *collaborative design network* (CDN), is proposed in (Xiang et al., 2004). Component-centered design is considered in the context of product lifecycle management (PLM) with the objective of overall optimal performance counting diverse uncertainty from materials, manufacturing and operating, as well as preference of vendors and users. A scheme for multiagent collaboration is developed (Xiang et al., 2005) which reduces complexity exponentially from that of a centralized design by exhaustive evaluation. This contribution goes further to enable efficient local design at each agent. Additional constraints for CDN are also proposed to facilitate model construction and verification.

## 2 Design Networks

As introduced above, this work is an essential part of the research towards multiagent collaborative design. Since the focus here is the decision process at a single agent, multiagent issues can be set aside without affecting the integrity of the result presented. In this section, we define a *design network* (DN) as the key graphical model associated with such an agent. The definition here extends that in (Xiang et al., 2004) so that it is not only more expressive in representing design knowledge, but also more restrictive to better guide model construction and verification. Aspects in (Xiang et al., 2004) are outlined here briefly for completeness. Readers are referred to reference for more details.

A design network is a triple $S = (V, E, P)$ whose structure is a connected DAG $G = (V, E)$. The set of nodes, each corresponds to a variable, is $V = D \cup T \cup M \cup U$. $D$ is an non-empty set of *design parameter*. $T$ is a set of *environmental factors* representing the uncertain manufacturing and operating environment for the product under design. $M$ is an non-

empty set of objective *performance measures* of the product. $U$ is an non-empty set of subjective *utility functions* of the designer.

$E$ is an non-empty set of *legal* arcs. We refer to distinct endpoints from $D$ as $d$ and $d'$, from $T$ as $t$ and $t'$, from $M$ as $m$ and $m'$, from $U$ as $u$ and $u'$, respectively. There are six types of legal arcs:

1. Arc $(d, d')$ signifies that the two parameters are involved in a design constraint.
2. Arc $(d, m)$ represents that performance $m$ depends on design parameter $d$.
3. Arc $(t, t')$ represents dependency between environmental factors. This type of arcs is added to legal arcs in (Xiang et al., 2004) to encode complex dependence relations among environmental factors.
4. Arc $(t, m)$ signifies that performance $m$ depends on environment factor $t$.
5. Arc $(m, m')$ defines $m'$ as a composite performance measure.
6. Arc $(m, u)$ signifies that utility $u$ depends on performance $m$.

$P$ is a set of potentials one associated with each node $x$ in the *form* of a conditional probability distribution $P(x|\pi(x))$, where $\pi(x)$ is the set of parent nodes of $x$. According to legal arcs, a design parameter $d$ can have only design parameters as parents. If $d$ is a root, $P(d)$ is a constant distribution. Otherwise, $P(d|\pi(d))$ must contain the value 0 (i.e., not strictly positive). The non-strictly-positive requirement can be understood as follows: When $\pi(d)$ is non-empty, $P(d|\pi(d))$ represents design constraints. It specifies under what configurations of $\pi(d)$, certain values of $d$ are illegal, and the value 0 signifies violation of design constraints.

Potential $P(t|\pi(t))$ is a typical probability distribution. According to legal arcs, $\pi(t)$ consists of environmental factors only. $P(m|\pi(m))$ is also a typical probability distribution, representing the uncertain dependence of the performance on design, environment, as well as other performance measures.

All utility variables are binary with domain $\{y, n\}$. The potential $P(u = y|\pi(u))$ is assigned a utility function $u(\pi(u))$ whose values range in $[0, 1]$ and include the two bounds. According to legal arcs, $\pi(u)$ consists of performance measures only. The potential $P(u = n|\pi(u))$ is assigned $1 - P(u = y|\pi(u))$. Each node $u$ is also assigned a weight $k \in [0, 1]$ such that the weights of all utility nodes sum to one.

With $P$ thus defined, $S$ is *syntactically* a Bayesian network. Assuming additive independence (Keeney and Raiffa, 1976) among utility variables, the expected utility of a design $\mathbf{d}$

$$EU(\mathbf{d}) = \sum_i k_i \ (\sum_{\mathbf{m}} u_i(\mathbf{m}) \ P(\mathbf{m}|\mathbf{d})) \qquad (1)$$

can be computed by standard probabilistic inference in $S$ (Xiang et al., 2004), where $\mathbf{d}$ (bold) is a configuration of $D$, $i$ indexes utility nodes in $U$, $\mathbf{m}$ (bold) is a configuration of the parents of $u_i$, and $k_i$ is the weight associated with $u_i$. Figure 1 shows a trivial example DN.

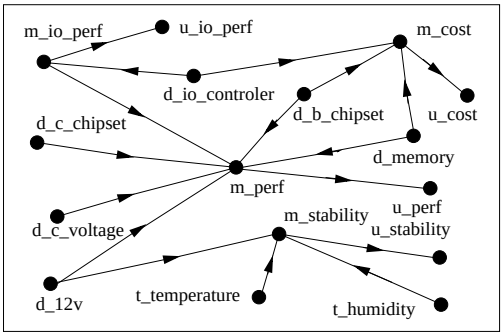



Figure 1: A trivial DN for PC motherboard.

Restriction of legal arcs does not constrain $S$ sufficiently so that every component is relevant to the design task, as will become clear below. We impose the following *essentiality* requirement to provide further guidance to model construction and to facilitate model verification.

For any design parameter $d$, if there exists a directed path in $S$ from $d$ to a utility node $u$, then $d$ is *essential*. Otherwise, $d$ is *non-essential*. If $d$ is non-essential, then the expected utility of any given design is independent of the value that $d$ takes. In other words, the optimal design over the remaining design parameters is independent of $d$, and the optimal

value for $d$ is undefined. Hence, $d$ can be removed from $S$ without affecting the optimal design over remaining design parameters and the maximum expected utility.

For any performance measure $m$, if there exists a directed path in $S$ from a design parameter $d$ to $m$, as well as a directed path from $m$ to a utility node $u$, then $m$ is *essential*. Without the path from $d$ to $m$, $m$ will not depend on any design. Without the path from $m$ to $u$, $m$ will not influence the optimal design. In either case, $m$ is deemed *non-essential* and can be removed without affecting the optimal design.

The case for environmental factors differs from the above. Consider a path $m \leftarrow t \rightarrow t'$, where $m$ is essential and $t'$ is a leaf. Suppose that the value of $t'$ is known at the time of design (an extension to the context for Eqn. (1)). Depending on the known value of $t'$, the impact of $t$ on $m$ may differ. If a directed path from each environmental factor to a utility node is required, the node $t'$ above will be disallowed. Therefore, an environmental factor $t$ is deemed *essential* if there exists an undirected path in $S$ from $t$ to a performance measure $m$. Otherwise, $t$ is *non-essential*.

We require that all design parameters, performance measures, and environmental factors in a design network to be essential.

## 3 Detecting Illegal Designs

An agent equipped with a DN can compute expected utility $EU(\mathbf{d})$ (Eqn. (1)) for each alternative design using standard probabilistic reasoning. However, finding the optimal design by exhaustively evaluating each design has the complexity $O(\kappa^{|D|})$, where $\kappa$ is the maximum number of values that a design parameter can take. We present a much more efficient method below.

An illegal design violates at least one design constraint. The sooner it can be detected, the sooner the evaluation computation can be directed to alternative designs and the more efficient the overall design computation. Suppose that a constraint involves a subset $X \subset D$ of binary design parameters $d_0, ..., d_j$, where $d_1, ..., d_j$ are the parents of $d_0$ in the DN, such that $d_0, ..., d_j$ cannot take value 0 all at the same time. This is specified in the DN by $P(d_0 = 0 | d_1 = 0, ..., d_j = 0) = 0$. Any design that are consistent with the configuration $(d_0 = 0, ..., d_j = 0)$ is an illegal design. Note that the number of such illegal designs are in the order of $O(k^{|D \setminus X|})$.

A DN is syntactically a Bayesian network and hence can be compiled into a junction tree (JT) $T$. Due to the moralization step in compilation, there exists a cluster $C \supseteq X$ in $T$. After compilation, each cluster $Q$ in $T$ is assigned a potential $B(Q)$ over the set $Q$ of member variables. The assignment to $C$ satisfies $B(d_0 = 0, ..., d_j = 0) = 0$.

First, we assume that $C = X$. Consider a different configuration of $C$ whose potential value is non-zero, for instance, the configuration $(d_0 = 0, d_1 = 1, d_2 = 0, ...., d_j = 0)$ such that $B(d_0 = 0, d_1 = 1, d_2 = 0, ..., d_j = 0) > 0$. To evaluate any design consistent with $(d_0 = 0, ..., d_j = 0)$, values $d_0 = 0$, ..., $d_j = 0$ are entered into $C$. When $d_1 = 0$ is entered into $C$, the potential value $B(d_0 = 0, d_1 = 1, d_2 = 0, ..., d_j = 0)$ is multiplied by 0 because the configuration is inconsistent with $d_1 = 0$. As the result, the updated potential value is $B(d_0 = 0, d_1 = 1, d_2 = 0, ..., d_j = 0) = 0$. To summarize, before entering values $d_1 = 0$, the potential value $B(d_0 = 0, ..., d_j = 0)$ is 0, and after entering $d_1 = 0$, the potential value for every other configuration is 0. Hence, after entering $d_0 = 0$, ..., $d_j = 0$, every potential value in $B(C)$ becomes 0.

Next, assume that $C = X \cup Y$, where $Y \neq \emptyset$ and $Y \cap X = \emptyset$. Since any potential $B(Y, d_0, ..., d_j)$ can be factorized into $B(Y, d_0, ..., d_j)B(d_0, ..., d_j)$, we conclude from the above that after entering $d_0 = 0$, ..., $d_j = 0$, every potential value in $B(C)$ becomes 0.

This analysis suggests a method to detect illegal designs. When a particular design (possibly illegal) is evaluated, for each cluster $C$ that has been assigned $P(d|\pi(d))$ relative to a design parameter $d$, enter the value of $d$ and the value of each design parameter in $\pi(d)$ into $C$. After entering, check if the sum of potential values of

$B(C)$ is 0. As soon as a positive test occurs, conclude that all designs that are consistent with this configuration of $(d, \pi(d))$ are illegal. There will be no need to evaluate any one of them.

## 4 Domain Division

In addition to early detection of illegal designs, we seek to evaluate legal designs more efficiently as follows. We assume that some separators in $T$ consist of design parameters only. The structure of an example DN is shown in Figure 2. It is converted into the JT in Figure 3. In the JT, three separators $\{d\_a, d\_c\}$, $\{d\_b, d\_c\}$ and $\{d\_a, d\_d\}$ consist of design parameters only.

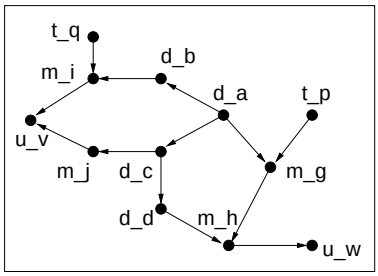



Figure 2: The structure of a design network.

Using these separators as boundaries, we compile the JT $T$ into a representation, called *division tree* that is more effective for design computation.

*Definition* 1. Let $S$ be a design network over an nonempty set of essential variables $V = D \cup T \cup M \cup U$. A `division tree` for $S$ is a tuple $\Gamma = (V, \Delta, \Theta, \Upsilon)$. $V$ is the `generating set` of $\Gamma$. $\Delta$ is a subset of the powerset $POW(V)$ of $V$ such that $\cup_{Q \in \Delta} = V$. Each element of $\Delta$ is called a `division`. $\Theta$ is composed of the following:

$$\Theta = \{< Q, Q' > | Q, Q' \in \Delta, Q \neq Q',$$

$$Q \cap Q' \neq \emptyset, Q \cap Q' \subset D\}.$$

Each unordered pair $< Q, Q' >$ is called a `separator` between the two divisions and is labeled by the intersection $Q \cap Q'$. $\Delta$ and $\Theta$ are so composed that $(V, \Delta, \Theta)$ forms a junction tree. $\Upsilon$ is a set of `division junction trees`. Its elements map one-to-one to elements of $\Delta$ such that each division junction tree has the corresponding division as its generating set.

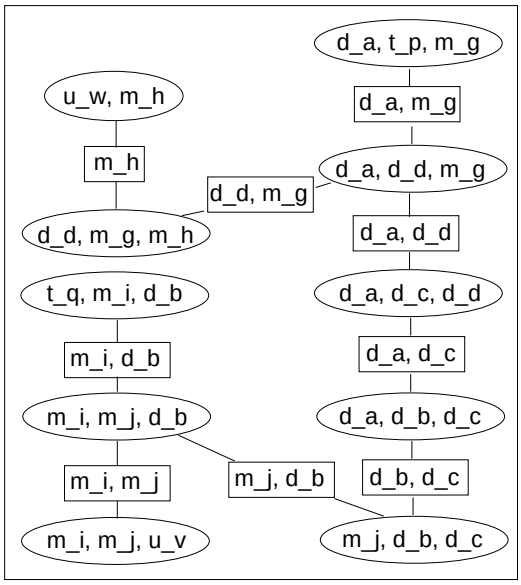



Figure 3: A junction tree representation of design network in Figure 2.

As an example, consider a division tree for the DN $S$ in Figure 2. The set $V$ of variables in $S$ is the generating set. The set of divisions is $\Delta = \{V_0, V_1, V_2, V_3\}$ as shown in Figure 4. The set $\Theta$ of division separators is also shown in the figure. It can be easily verified that the graph depicted is a JT. The set of division JTs is $\Upsilon = \{ST_0, ST_1, ST_2, ST_3\}$ as shown in Figure 5, where the generating set of division JT $ST_i$ is division $V_i$.

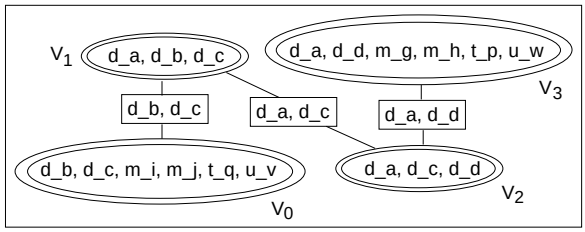



Figure 4: The division tree for design network in Figure 2. Each double-oval represents a division. Each box represents a separator.

Once DN $S$ has been compiled into JT $T$, construction of a division tree $\Gamma$ is straightforward: The set of division JTs are obtained from $T$ by removing its separators that are composed of design parameters only. For instance, after removing separators $\{d\_a, d\_c\}$, $\{d\_b, d\_c\}$ and

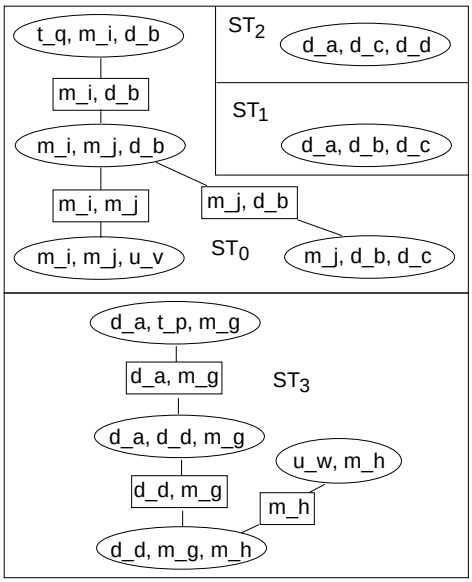



Figure 5: Division JTs for division tree in Figure 4. $ST_i$ corresponds to division $V_i$.

$\{d\_a, d\_d\}$, $T$ is split into division JTs in Figure 5. The generating sets of these division JTs become divisions in $\Delta$. The removed separators become division separators in $\Theta$. It is a simple matter to show that $(V, \Delta, \Theta)$ thus constructed forms a junction tree.

## 5 Design Evaluation within Division

Each division $V_i$ contains an nonempty subset $D_i$ of design parameters, where $i$ indexes the division. This is true since the division contains at least the design parameters that form the separator between itself and another division.

First, for each configuration of $D_i$, whether it is a legal partial design needs to be determined, to the extent allowed by information available in the division. This can be achieved as outlined in Section 3. A division may or may not contain utility variables. If it does not contain utility variables, then the evaluation within the division is complete. For the division tree in Figure 4, $V_1$ and $V_2$ are such divisions.

If the division contains utility variables, then for each legal configuration of $D_i$, further evaluation needs to be performed. This can be done by belief propagation within the division JT. The expected utility for each utility function can then be retrieved from the corresponding utility variable. The expected utility contribution of the partial design can be obtained by combining the result from individual utility variables.

More precisely, in division $V_i$, the following is obtained for each configuration $\mathbf{d_i}$ of $D_i$,

$$EU(\mathbf{d_i}) = \sum_j k_j \ (\sum_{\mathbf{m}} u_j(\mathbf{m}) \ P(\mathbf{m}|\mathbf{d_i}))$$

where $j$ indexes utility nodes in $V_i$, $\mathbf{m}$ is a configuration of the parents of $u_j$, and $k_j$ is the weight associated with $u_j$.

We extend the definition of $EU(\mathbf{d_i})$ to divisions without utility variables and to illegal partial design $\mathbf{d_i}$ as follows: We set $EU(\mathbf{d_i}) = 0$ if division $V_i$ contains no utility variables. We set $EU(\mathbf{d_i}) = null$ if $\mathbf{d_i}$ is illegal.

After the above evaluation on each configuration of $D_i$, the design evaluation within the division is complete.

## 6 Message Passing in Division Tree

To combine design evaluations at individual divisions and determine the optimal design, messages are passed between divisions, along the separators of the division tree (e.g., Figure 4). The message passing is organized into three rounds along the division tree. The syntax and semantics of messages in each round differ.

The algorithms that we present below are mostly executed by individual divisions, except one by the agent $A$. Without losing generality, we denote the division object executing the algorithm by $V_0$. The execution is activated by a caller, denoted by $V_c$, which is either an adjacent division of $V_0$ in $\Gamma$ or agent $A$. The separator between caller division $V_c$ and $V_0$ is denoted as $R_c$. If $V_0$ has additional adjacent divisions, they are denoted as $V_1, V_2, ..., V_z$ and their separators with $V_0$ are denoted as $R_1, R_2, ..., R_z$, respectively.

## 7 Collecting Utility Contribution

In the first round of message passing, division $V_0$ receives a vector message from each adjacent division $V_i$. Elements of the vector are indexed by

partial designs over $R_i$. The $k$th element of the vector, indexed by partial design $\mathbf{e_i^k}$, is denoted as $MEV_i^k$. The corresponding components for the vector message that $V_0$ sends to division $V_c$ are $\mathbf{e_c^k}$ (over $R_c$) and $MEV_c^k$, respectively.

If $V_i$ is a leaf on the division tree (whose only adjacent division is $V_0$), $MEV_i^k$ corresponds to the maximum expected utility

$$MEV_i^k = max_{\mathbf{d_j}} EU(\mathbf{d_i})$$

over all partial design $\mathbf{d_i}$ that are consistent with $\mathbf{e_i^k}$. Otherwise, its interpretation becomes clear below.

The following algorithm, when executed by each division, propagates utility contributions of partial designs at individual divisions inwards along division tree. The vector message sent from $V_i$ to $V_0$ is denoted as $MEV_i$ and that sent from $V_0$ to $V_c$ is denoted as $MEV_c$.

*Algorithm* 1 *(CollectDivisionUtility).* When division $V_0$ is called by $V_c$ to CollectDivisionUtility, it does the following:

1. For each adjacent division $V_i$, $V_0$ calls CollectDivisionUtility in $V_i$ and receives $MEV_i$ from $V_i$.

2. For each partial design $\mathbf{d_0}$, update

$$EU(\mathbf{d_0}) = EU(\mathbf{d_0}) + \sum_i MEV_i^k,$$

   where $MEV_i^k$ is indexed by partial design $\mathbf{e_i^k}$ and $\mathbf{e_i^k}$ is consistent with $\mathbf{d_0}$.

3. If $V_c$ is an adjacent division, for each partial design $\mathbf{e_c^k}$, $V_0$ computes

$$MEV_c^k = max_{\mathbf{d_0}} EU(\mathbf{d_0})$$

   over all partial design $\mathbf{d_0}$ that are consistent with $\mathbf{e_c^k}$, labels one such partial design that reaches the value $MEV_c^k$ by $\mathbf{d_c^{k*}}$ (breaking ties arbitrarily), and sends $MEV_c$ to $V_c$.

Note that $EU(\mathbf{d_0})$ may be equal to *null*. When *null* participates in an addition, we require that the sum is *null*. When *null* participates in a *max* operation, we require that the result is *null* if and only if all operants are *null*.

## 8 Distributing Optimal Design

In the second round of message passing, division $V_0$ receives from division $V_c$ a partial design $\mathbf{e_c^k}$ that is consistent with the optimal design. Combining it with the design evaluation performed earlier within the division, $V_0$ identifies the optimal partial design within the division. It then sends the optimal partial design relative to the separator of each adjacent division to the corresponding $V_i$. As message passing progresses, each division identifies the optimal partial design within the division. This is achieved by the following algorithm.

*Algorithm* 2 *(DistributeOptimalDivisionDesign).* When division $V_0$ is called by $V_c$ to DistributeOptimalDivisionDesign, it does the following:

1. If $V_c$ is an adjacent division, $V_0$ receives a partial design $\mathbf{e_c^k}$ over $R_c$ from $V_c$. Then, among partial designs over $D_0$ that are consistent with $\mathbf{e_c^k}$, it identifies the one with the highest $EU$ value (breaking ties arbitrarily) and label it as $\mathbf{d_0^*}$.

2. Otherwise, among all partial designs over $D_0$, it identifies the one with the highest $EU$ value (breaking ties arbitrarily) and label it as $\mathbf{d_0^*}$.

3. For each adjacent division $V_i$, call DistributeOptimalDivisionDesign in $V_i$ and send the partial design $\mathbf{e_i^k}$ that is consistent with $\mathbf{d_0^*}$ to $V_i$.

## 9 Collecting Optimal Design

In the last round of message passing, division $V_0$ receives the optimal partial design over $D_i$ from each adjacent division $V_i$. It combines them with its own optimal partial design over $D_0$ and sends the result to division $V_c$. At the end of this round, the optimal design over $D$ is obtained.

*Algorithm* 3 *(CollectOptimalDesign).* When division $V_0$ is called by $V_c$ to CollectOptimalDesign, it does the following:

1. For each adjacent division $V_i$, $V_0$ calls CollectOptimalDesign in $V_i$ and receives $\mathbf{d_i^*}$ from $V_i$.

2. Combine $\mathbf{d_0^*}$ with all $\mathbf{d_i^*}$ received and send result to $V_c$.

The following algorithm activates the three rounds of message passing in turn and is executed by the agent $A$.

*Algorithm* 4 *(OptimalDesignByDivisionTree).*

1. Select a division $V_x$ arbitrarily.
2. Call CollectDivisionUtility in $V_x$.
3. Call DistributeOptimalDivisionDesign in $V_x$.
4. Call CollectOptimalDesign in $V_x$.
5. When $V_x$ finishes, receive from it design $\mathbf{d}$ over $D$.

Soundness of OptimalDesignByDivisionTree is established below. Proposition 2 asserts that $\mathbf{d}$ produced in the algorithm is a legal design. Only a proof sketch is given due to space limit.

*Proposition* 2. The design $\mathbf{d}$ produced by OptimalDesignByDivisionTree is legal.

Proof sketch: View the division tree as rooted at $V_x$ and use induction on its depth *dep*. The base case is $dep = 0$. There is a single division and the proposition can be easily shown. Assume the proposition for $dep \leq m$ and consider $dep = m+1$. Let the root division be $V_0$ and its adjacent divisions be $V_i$ $(i = 1, ..., z)$. The subtree rooted at each $V_i$ has a depth $\leq m$. By assumption, if CollectDivisionUtility is called on each $V_i$ by the agent, followed by a call of DistributeOptimalDivisionDesign on $V_i$, followed by a call of CollectOptimalDesign on $V_i$, then the (partial) design returned by $V_i$ is legal relative to the subtree.

The actual execution differs from the above as follows: CollectDivisionUtility is called on each $V_i$ by $V_0$ which receives $MEV_i$ from $V_i$. For each element $MEV_i^k$ in $MEV_i$, it is *null* if there exists no legal (partial) design consistent with $\mathbf{e_i^k}$ in the subtree rooted at $V_i$. By null addition, any partial design $\mathbf{d_0}$ in $V_0$ consistent with $\mathbf{e_i^k}$ will be evaluated to $EU(\mathbf{d_0}) = null$. Hence, $\mathbf{d_0}$ cannot be selected as $\mathbf{d_0^*}$ by $V_0$ during DistributeOptimalDivisionDesign and cannot be part of the design returned through CollectOptimalDesign. [end of sketch]

Theorem 3 shows that $\mathbf{d}$ is an optimal design.

*Theorem* 3. The design $\mathbf{d}$ produced by OptimalDesignByDivisionTree is optimal.

Proof sketch: It suffices to show that $EU(\mathbf{d_0^*})$ obtained by the root division $V_0$ in step 2 of DistributeOptimalDivisionDesign is the maximum expected utility over all legal designs. Once this is established, it follows that a design, that attains this maximum expected utility and is restricted to each division, is $\mathbf{d_0^*}$ labeled by the corresponding division during DistributeOptimalDivisionDesign. CollectOptimalDesign simply assembles them together. The body of the proof uses induction with a structure similar to the proof above. [end of sketch]

## 10 Complexity

Denote the total number of design parameters by $|D|$ and the maximum number of possible values of a design parameter by $\kappa$. A centralized optimal design that evaluates all designs exhaustively has the complexity $O(\kappa^{|D|})$.

For OptimalDesignByDivisionTree, let the number of divisions be $|\Delta|$, the maximum number of design parameters per division be $\delta$, and the maximum cardinality of division separators be $q$. During CollectDivisionUtility, each division evaluates $O(\kappa^\delta)$ partial designs and sends a message of size $O(\kappa^q)$ to the caller. Hence, the complexity of OptimalDesignByDivisionTree is $O(|\Delta|\ \kappa^\delta + (|\Delta| - 1)\ \kappa^q)$. Normally, $q$ is much smaller than $\delta$ and the complexity becomes $O(|\Delta|\ \kappa^\delta)$. When $\delta$ is upper-bounded, OptimalDesignByDivisionTree is efficient.

## 11 Other Related Work

In addition to previous work reviewed in Section 1, we discuss relations of this contribution to other related work below:

In the literature on graphical models, a related work is strong junction tree (Jensen et al., 1994) which addresses the issue of sequential decision making. As indicated by Paredis

et al. (Paredis et al., 2006), point-based design corresponds to sequential decision making. Instead, the optimal design addressed in this contribution takes the approach of set-based design, which involves simultaneous evaluation of a much large number of design parameters than what is considered in a typical sequential decision problem. The issue of design constraint violation is also dealt with in this contribution.

Another related work is nested junction trees (Kjaerulff, 1997), where a JT is nested in a cluster of a higher level JT to reduce space complexity and to allow more efficient belief propagation. The division tree, proposed in this contribution, is also a nested JT structure. The computation conducted within a division, however, goes beyond probabilistic reasoning. It reasons about design decisions, constraints and utilities, and is decision-theoretic in nature.

In the literature on constraint satisfaction problem (CSP), constraint graphs are converted to JTs to solve CSPs (Dechter and Pearl, 1989). The current contribution essentially extends that approach to constraint optimization and with a decision-theoretic objective function.

## 12 Conclusion

We refined the definition for design networks regarding legal arcs and essentiality. The new definition improves expressiveness and guidance to model construction and verification. We compiled a design network into a division tree and presented algorithms that combine probabilistic, constraint-based and decision-theoretic reasoning for optimal design in the compiled structure. This result not only provides a computational mechanism for single-agent optimal design, but also fills in a gap in optimal collaborative design.

The presented algorithm suite solves the problem of decision-theoretic optimal design in the context of catalog design where discrete design options are configured. The algorithm suite derives its efficiency by decomposing the design domain according to design separators in division trees. When multiple optimal designs exist, the algorithm suite returns one of them arbitrarily, but can be extended to return all.

## Acknowledgments

Financial support to this research is provided by NSERC of Canada.

## References

R.D. Braun, I.M. Kroo, and A.A. Moore. 1996. Use of the collaborative optimization architecture for launch vehicle design. In *Proc. 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages 306–318.

R. Dechter and J. Pearl. 1989. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366.

F. Jensen, F.V. Jensen, and S.L. Dittmer. 1994. From influence diagrams to junction trees. In *Proc. 10th Conf. on Uncertainty in Artificial Intelligence*, pages 367–373.

R.L. Keeney and H. Raiffa. 1976. *Decisions with Multiple Objectives*. Cambridge.

U. Kjaerulff. 1997. Nested junction trees. In *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*, pages 294–301, Providence, Rhode Island.

G. Konduri and A. Chandrakasan. 1999. A framework for collaborative and distributed web-based design. In *Proc. 36th Design Automation Conference*, pages 898–903.

C. Paredis, J. Aughenbaugh, R. Malak, and S. Rekuc. 2006. Set-based design: a decision-theoretic perspective. In *Proc. Frontiers in Design & Simulation Research 2006 Workshop*, pages 1–25.

D.K. Sobek, A.C. Ward, and J.K. Liker. 1999. Toyota's principles of set-based concurrent engineering. *Sloan Management Review*, 40(2):67–84.

A.C. Ward. 1989. *A Theory of Quantitative Inference Applied to A Mechanical Design Compiler*. Ph.D. thesis, MIT, Department of Mechanical Engineering.

Y. Xiang, J. Chen, and A. Deshmukh. 2004. A decision-theoretic graphical model for collaborative design on supply chains. In A.Y. Tawfik and S.D. Goodwin, editors, *Advances in Artificial Intelligence, LNAI 3060*, pages 355–369. Springer.

Y. Xiang, J. Chen, and W.S. Havens. 2005. Optimal design in collaborative design network. In *Proc. 4th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'05)*, pages 241–248.

# Hybrid Loopy Belief Propagation

**Changhe Yuan**
Department of Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
cyuan@cse.msstate.edu

**Marek J. Druzdzel**
Decision Systems Laboratory
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15260
marek@sis.pitt.edu

**Abstract**

We propose an algorithm called *Hybrid Loopy Belief Propagation* (HLBP), which extends the *Loopy Belief Propagation* (LBP) (Murphy et al., 1999) and *Nonparametric Belief Propagation* (NBP) (Sudderth et al., 2003) algorithms to deal with general hybrid Bayesian networks. The main idea is to represent the LBP messages with mixture of Gaussians and formulate their calculation as Monte Carlo integration problems. The new algorithm is general enough to deal with hybrid models that may represent linear or nonlinear equations and arbitrary probability distributions.

## 1 Introduction

Some real problems are more naturally modelled by hybrid Bayesian networks that contain mixtures of discrete and continuous variables. However, several factors make inference in hybrid models extremely hard. First, linear or nonlinear deterministic relations may exist in the models. Second, the models may contain arbitrary probability distributions. Third, the orderings among the discrete and continuous variables may be arbitrary. Since the general case is difficult, existing research often focuses on special instances of hybrid models, such as *Conditional Linear Gaussians* (CLG) (Lauritzen, 1992). However, one major assumption behind CLG is that discrete variables cannot have continuous parents. This limitation was later addressed by extending CLG with *logistic* and *softmax functions* (Lerner et al., 2001; Murphy, 1999). More recent research begin to develop methodologies for more general non-Gaussian models, such as *Mixture of Truncated Exponentials* (MTE) (Moral et al., 2001; Cobb and Shenoy, 2005), and *junction tree algorithm with approximate clique potentials* (Koller et al., 1999). However, most of these approaches rely on the junction tree algorithm (Lauritzen and Spiegelhalter, 1988). As Lerner et al. (Lerner et al., 2001) pointed out, it is important to have alternative solutions in case that junction tree algorithm-based methods are not feasible.

In this paper, we propose the *Hybrid Loopy Belief Propagation* algorithm, which extends the *Loopy Belief Propagation* (LBP) (Murphy et al., 1999) and *Nonparametric Belief Propagation* (NBP) (Sudderth et al., 2003) algorithms to deal with general hybrid Bayesian networks. The main idea is to represent LBP messages as *Mixtures of Gaussians* (MG) and formulate their calculation as Monte Carlo integration problems. The extension is far from trivial due to the enormous complexity brought by deterministic equations and mixtures of discrete and continuous variables. Another advantage of the algorithm is that it approximates the true posterior probability distributions, unlike most existing approaches which only produce their first two moments for CLG models.

## 2 Hybrid Loopy Belief Propagation

To extend LBP to hybrid Bayesian networks, we need to know how to calculate the LBP messages in hybrid models. There are two kinds of messages defined in LBP. Let $X$ be a node in a hybrid model. Let $Y_j$ be $X$'s children and $U_i$

be $X$'s parents. The $\lambda_X^{(t+1)}(u_i)$ message that $X$ sends to its parent $U_i$ is given by:

$$\lambda_X^{(t+1)}(u_i) = \alpha \int_x \lambda_X(x) \prod_j \lambda_{Y_j}^{(t)}(x) \int_{u_k:k\neq i} P(x|u) \prod_{k\neq i} \pi_X^{(t)}(u_k) \quad (1)$$

and the message $\pi_{Y_j}^{(t+1)}(x)$ that $X$ sends to its child $Y_j$ is defined as:

$$\pi_{Y_j}^{(t+1)}(x) = \alpha\lambda_X(x) \prod_{k\neq j} \lambda_{Y_k}^{(t)}(x) \int_u P(x|u) \prod_k \pi_X^{(t)}(u_k) \quad (2)$$

where $\lambda_X(x)$ is a message that $X$ sends to itself representing evidence. For simplicity, I use only integrations in the message definitions. It is evident that no closed-form solutions exist for the messages in general hybrid models. However, we observe that their calculations can be formulated as Monte Carlo integration problems.

First, let us look at the $\pi_{Y_j}^{(t+1)}(x)$ message defined in Equation 2. We can rearrange the equation in the following way:

$$\pi_{Y_j}^{(t+1)}(x) = \alpha \int_u \left\{ \overbrace{\lambda_X(x) \prod_{k\neq j} \lambda_{Y_k}^{(t)}(x) P(x|u) \prod_k \pi_X^{(t)}(u_k)}^{P(x,u)} \right\} .$$

Essentially, we put all the integrations outside of the other operations. Given the new formula, we realize that we have a joint probability distribution over $x$ and $u_i$s, and the task is to integrate all the $u_i$s out and get the marginal probability distribution over $x$. Since $P(x,u)$ can be naturally decomposed into $P(x|u)P(u)$, the calculation of the message can be solved using a Monte Carlo sampling technique called *Composition method* (Tanner, 1993). The idea is to first draw samples for each $u_i$s from $\pi_X^{(t)}(u_i)$, and then sample from the product of $\lambda_X(x) \prod_{k\neq j} \lambda_{Y_k}^{(t)}(x) P(x|u)$. We will discuss how to take the product in the next subsection. For now, let us assume that the computation is possible. To make life even easier, we make further modifications and get

$$\pi_{Y_j}^{(t+1)}(x) = \alpha \int_u \left\{ \frac{\lambda_X(x) \prod_k \lambda_{Y_k}^{(t)}(x) P(x|u) \prod_k \pi_X^{(t)}(u_k)}{\lambda_{Y_j}^{(t)}(x)} \right\} .$$

Now, for the messages sent from $X$ to its different children, we can share most of the calculation. We first get samples for $u_i$s, and then sample from the product of $\lambda_X(x) \prod_k \lambda_{Y_k}^{(t)}(x) P(x|u)$. For each different message $\pi_{Y_j}^{(t+1)}(x)$, we use the same sample $x$ but assign it different weights $1/\lambda_{Y_i}^{(t)}(x)$.

Let us now consider how to calculate the $\lambda_X^{(t+1)}(u_i)$ message defined in Equation 1. First, we rearrange the equation analogously:

$$\lambda_X^{(t+1)}(u_i) = \alpha \int_{x,u_k:k\neq i} \left\{ \overbrace{\lambda_X(x) \prod_j \lambda_{Y_j}^{(t)}(x) P(x|u) \prod_{k\neq i} \pi_X^{(t)}(u_k)}^{P(x,u_k:k\neq i|u_i)} \right\} \quad (3)$$

It turns out that here we are facing a quite different problem from the calculation of $\pi_{Y_j}^{(t+1)}(x)$. Note that now we have $P(x, u_k : k \neq i|u_i)$, a joint distribution over $x$ and $u_k (k \neq i)$ conditional on $u_i$, so the whole expression is only a likelihood function of $u_i$, which is not guaranteed to be integrable. As in (Sudderth et al., 2003; Koller et al., 1999), we choose to restrict our attention to densities and assume that the ranges of all continuous variables are bounded (maybe large). The assumption only solves part of the problem. Another difficulty is that composition method is no longer applicable here, because we have to draw samples for all parents $u_i$ before we can decide $P(x|u)$. We note that for any fixed $u_i$, Equation 3 is an integration over $x$ and $u_k, k \neq i$ and can be estimated using Monte Carlo methods. That means that we can estimate $\lambda_X^{(t+1)}(u_i)$ up to a constant for any value $u_i$, although we do not know how to sample from it. This is exactly the time when importance sampling becomes handy. We can estimate the message by drawing a set of importance samples as follows: sample $u_i$ from a

chosen importance function, estimate $\lambda_X^{(t+1)}(u_i)$ using Monte Carlo integration, and assign the ratio between the estimated value and $I(u_i)$ as the weight for the sample. A simple choice for the importance function is the uniform distribution over the range of $u_i$, but we can improve the accuracy of Monte Carlo integration by choosing a more informed importance function, the corresponding message $\lambda_X^{(t)}(u_i)$ from the last iteration. Because of the iterative nature of LBP, messages usually keep improving over each iteration, so they are clearly better importance functions.

Note that the preceding discussion is quite general. The variables under consideration can be either discrete or continuous. We only need to know how to sample from or evaluate the conditional relations. Therefore, we can use any representation to model the case of discrete variables with continuous parents. For instance, we can use general softmax functions (Lerner et al., 2001) for the representation.

We now know how to use Monte Carlo integration methods to estimate the LBP messages, represented as sets of weighted samples. To complete the algorithm, we need to figure out how to propagate these messages. Belief propagation involves operations such as sampling, multiplication, and marginalization. Sampling from a message represented as a set of weighted samples may be easy to do; we can use resampling technique. However, multiplying two such messages is not straightforward. Therefore, we choose to use density estimation techniques to approximate each continuous message using a *mixture of Gaussians* (MG) (Sudderth et al., 2003). A $K$ component mixture of Gaussian has the following form

$$M(x) = \sum_{i=1}^{K} w_i N(x; \mu_i, \sigma_i) \ , \qquad (4)$$

where $\sum_{i=1}^{K} w_i = 1$. MG has several nice properties. First, we can approximate any continuous distribution reasonably well using a MG. Second, it is closed under multiplication. Last, we can estimate a MG from a set of weighted samples using a regularized version of *expectation maximization* (EM) (Dempster et al., 1977; Koller et al., 1999) algorithm.

**Algorithm**: HLBP

1. Initialize the messages that evidence nodes send to themselves and their children as indicating messages with fixed values, and initialize all other messages to be uniform.
2. **while** (stopping criterion not satisfied)
   Recompute all the messages using Monte Carlo integration methods.
   Normalize discrete messages.
   Approximate all continuous messages using MGs.
   **end while**
3. Calculate $\lambda(x)$ and $\pi(x)$ messages for each variable.
4. Calculate the posterior probability distributions for all the variables by sampling from the product of $\lambda(x)$ and $\pi(x)$ messages.

Figure 1: The Hybrid Loopy Belief Propagation algorithm.

Given the above discussion, we outline the final HLBP algorithm in Figure 1. We first initialize all the messages. Then, we iteratively recompute all the messages using the methods described above. In the end, we calculate the messages $\lambda(x)$ and $\pi(x)$ for each node $X$ and use them to estimate the posterior probability distribution over $X$.

## 3 Product of Mixtures of Gaussians

One question that remains to be answered in the last section is how to sample from the product of $\lambda_X(x) \prod_{k \neq j} \lambda_{Y_k}^{(t)}(x) P(x|u)$. We address the problem in this section. If $P(x|u)$ is a continuous probability distribution, we can approximate $P(x|u)$ with a MG. Then, the problem becomes how to compute the product of several MGs. The approximation is often a reasonable thing to do because we can approximate any continuous probability distribution reasonably well using MG. Even when such approximation is poor, we can approximate the product of $P(x|u)$ with an MG using another MG. One example is that the product of Gaussian distribu-

tion and logistic function can be approximated well with another Gaussian distribution (Murphy, 1999).

Suppose we have messages $M_1, M_2, ..., M_K$, each represented as an MG. Sudderth et al. use Gibbs sampling algorithm to address the problem (Sudderth et al., 2003). The shortcoming of the Gibbs sampler is its efficiency: We usually have to carry out several iterations of Gibbs sampling in order to get one sample.

Here we propose a more efficient method based on the chain rule. If we treat the selection of one component from each MG message as a random variable, which we call a *label*, our goal is to draw a sample from the joint probability distribution of all the labels $P(L_1, L_2, ..., L_K)$. We note that the joint probability distribution of the labels of all the messages $P(L_1, L_2, ..., L_K)$ can be factorized using the chain rule as follows:

$$P(L_1, L_2, ..., L_K) = P(L_1) \prod_{i=2}^{K} P(L_i | L_1, ..., L_{i-1}) \ . \tag{5}$$

Therefore, the idea is to sample from the labels sequentially based on the prior or conditional probability distributions. Let $w_{ij}$ be the weight for the $j$th component of $i$th message and $\mu_{ij}$ and $\sigma_{ij}$ be the component's parameters. We can sample from the product of messages $M_1, ..., M_K$ using the algorithm presented in Figure 2. The main idea is to calculate the conditional probability distributions cumulatively. Due to the Gaussian densities, the method has to correct the bias introduced during the sampling by assigning the samples weights. The method only needs to go over the messages once to obtain one importance sample and is more efficient than the Gibbs sampler in (Sudderth et al., 2003). Empirical results show that the precision obtained by the importance sampler is comparable to the Gibbs sampler given a reasonable number of samples.

**Algorithm**: Sample from a product of MGs $M_1 \times M_2 \times ... \times M_K$.

1. Randomly pick a component, say $j_1$, form the first message $M_1$ according to its weights $w_{11}, ..., w_{1J_1}$.
2. Initialize cumulative parameters as follows
   $\mu_1^* \leftarrow \mu_{1j_1}$; $\sigma_1^* \leftarrow \sigma_{1j_1}$.
3. $i \leftarrow 2$; $wImportance \leftarrow w_{1j_1}$.
4. **while** $(i \leq K)$

   Compute new parameters for each component of $ith$ message as follows
   $\hat{\sigma}_{ik}^* \leftarrow ((\sigma_{i-1}^*)^{-1} + (\sigma_{ik})^{-1})^{-1}$,
   $\hat{\mu}_{ik}^* \leftarrow (\frac{\mu_{ik}}{\sigma_{ik}} + \frac{\mu_{i-1}^*}{\sigma_{i-1}^*})\hat{\sigma}_{ik}^*$.

   Compute new weights for $ith$ message with any $x$
   $\hat{w}_{ik}^* = w_{ik} \hat{w}_{i-1 j_{i-1}}^* \frac{N(x;\mu_{i-1}^*, \sigma_{i-1}^*) N(x;\mu_{ik}, \sigma_{ik})}{N(x;\hat{\mu}_{ik}^*, \hat{\sigma}_{ik}^*)}$.

   Calculate the normalized weights $\bar{w}_{ik}^*$.

   Randomly pick a component, say $j_i$, from the $ith$ message using the normalized weights.

   $\mu_i^* \leftarrow \hat{\mu}_{ij_i}^*$; $\sigma_i^* \leftarrow \hat{\sigma}_{ij_i}^*$.

   $i \leftarrow i + 1$;

   $wImportance = wImportance \times \bar{w}_{ij_i}^*$.

   **end while**
5. Sample from the Gaussian with mean $\mu_K^*$ and variance $\sigma_K^*$.
6. Assign the sample weight $\hat{w}_{Kj_K}^* / wImportance$.

Figure 2: Sample from a product of MGs.

## 4 Belief Propagation with Evidence

Special care is needed for belief propagation with evidence and deterministic relations. In our preceding discussion, we approximate $P(x|u)$ using MG if $P(x|u)$ is a continuous probability distribution. It is not the case if $P(x|u)$ is deterministic or if $X$ is observed. We discuss the following several scenarios separately.

*Deterministic relation without evidence*: We simply evaluate $P(x|u)$ to get the value $x$ as a sample. Because we did not take into account the $\lambda$ messages, we need to correct our bias using weights. For the message $\pi_{Y_i}(x)$ sent from $X$ to its child $Y_i$, we take $x$ as the sample and assign it weight $\lambda_X(x) \prod_{k \neq i} \lambda_{Y_k}^{(t)}(x)$. For the message $\lambda_X(u_i)$ sent from $X$ to its parent $U_i$, we take value $u_i$ as a sample for $U_i$ and assign it weight $\lambda_X(x) \prod_k \lambda_{Y_k}^{(t)}(x) / \lambda_X^{(t)}(u_i)$.

*Stochastic relation with evidence*: The messages $\pi_{Y_j}(x)$ sent from evidence node $X$ to its

children are always indicating messages with fixed values. The messages $\lambda_{Y_j}(x)$ sent from the children to $X$ have no influence on $X$, so we need not calculate them. We only need to update the messages $\lambda_X(u_i)$, for which we take the value $u_i$ as the sample and assign it weight $\lambda_X(e)\prod_k \lambda^{(t)}_{Y_k}(e)/\lambda^{(t)}_X(u_i)$, where $e$ is the observed value of $X$.

*Deterministic relation with evidence*: This case is the most difficult. To illustrate this case more clearly, we use a simple hybrid Bayesian network with one discrete node $A$ and two continuous nodes $B$ and $C$ (see Figure 3).

| $a$ | 0.7 |
|---|---|
| $\neg a$ | 0.3 |

| $B$ |
|---|
| $N(B;1,1)$ |

| $P(C\|A,B)$ | $a$ | $\neg a$ |
|---|---|---|
| $C$ | $C=B$ | $C=2*B$ |

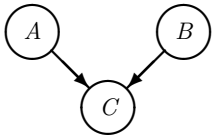



Figure 3: A simple hybrid Bayesian network.

**Example 1.** Let $C$ be observed at state 2.0. Given the evidence, there are only two possible values for $B$: 2.0 when $A = a$ and 1.0 when $A = \neg a$. We need to calculate messages $\lambda_C(a)$ and $\lambda_C(b)$. If we follow the routine and sample from $A$ and $B$ first, it is extremely unlikely for us to hit a feasible sample; Almost all the samples that we get would have weight 0.0. Clearly we need a better way to do that.

First, let us consider how to calculate the message $\lambda_C(a)$ sent from $C$ to $A$. Suppose we choose uniform distribution as the importance function, we first randomly pick a state for $A$. After the state of $A$ is fixed, we note that we can solve $P(C|A,B)$ to get the state for $B$. Therefore, we need not sample for $B$ from $\pi_C(b)$, in this case $N(B;1,1)$. We then assign $N(B;1,1)$ as weight for the sample. Since $A$'s two states are equally likely, the message $\lambda_C(A)$ would be proportional $\{N(2;1,1), N(1;1,1)\}$.

For the message $\lambda_C(b)$ message sent from $C$ to $B$, since we know that $B$ can only take two values, we choose a distribution that puts equal probabilities on these two values as the importance function, from which we sample for $B$. The state of $B$ will also determine $A$ as follows: when $B = 2$, we have $A = a$; when $B = 1$, we have $A = \neg a$. We then assign weight 0.7 to the sample if $B = 2$ and 0.3 if $B = 1$. However, the magic of knowing feasible values for $B$ is impossible in practice. Instead, we first sample for $A$ from $\pi_C(A)$, in this case, $\{0.7, 0.3\}$, given which we can solve $P(C|A,B)$ for $B$ and assign each sample weight 1.0. So $\lambda_C(b)$ have probabilities proportional to $\{0.7, 0.3\}$ on two values 2.0 and 1.0.

In general, in order to calculate $\lambda$ messages sent out from a deterministic node with evidence, we need to sample from all parents except one, and then solve $P(x|u)$ for that parent. There are several issues here. First, since we want to use the values of other parents to solve for the chosen parent, we need an equation solver. We used an implementation of the *Newton's method for solving nonlinear set of equations* (Kelley, 2003). However, not all equations are solvable by this equation solver or any equation solver for that matter. We may want to choose the parent that is easiest to solve. This can be tested by means of a preprocessing step. In more difficult cases, we have to resort to users' help and ask at the model building stage for specifying which parent to solve or even manually specify the inverse functions. When there are multiple choices, one heuristic that we find helpful is to choose the continuous parent with the largest variance.

## 5 Lazy LBP

We can see that HLBP involves repeated density estimation and Monte Carlo integration, which are both computationally intense. Efficiency naturally becomes a concern for the algorithm. To improve its efficiency, we propose a technique called *Lazy LBP*, which is also applicable to other extensions of LBP. The tech-

nique serves as a summarization that contains both my original findings and some commonly used optimization methods.

After evidence is introduced in the network, we can pre-propagate the evidence to reduce computation in HLBP. First, we can plug in any evidence to the conditional relations of its children, so we need not calculate the $\pi$ messages from the evidence to its children. We need not calculate the $\lambda$ messages from its children to the evidence either. Secondly, evidence may determine the value of its neighbors because of deterministic relations, in which case we can evaluate the deterministic relations in advance so that we need not calculate messages between them.

Furthermore, from the definitions of the LBP messages, we can easily see that we do not have to recompute the messages all the time. For example, the $\lambda(x)$ messages from the children of a node with no evidence as descendant are always uniform. Also, a message needs not be updated if the sender of the message has received no new messages from neighbors other than the recipient.

Based on Equation 1, $\lambda$ messages should be updated if incoming $\pi$ messages change. However, we have the following result.

**Theorem 1.** *The $\lambda$ messages sent from a non-evidence node to its parents remain uniform before it receives any non-uniform messages from its children, even though there are new $\pi$ messages coming from the parents.*

*Proof.* Since there are no non-uniform $\lambda$ messages coming in, Equation 1 simplifies to

$$\begin{aligned}\lambda_X^{(t+1)}(u_i) &= \alpha \int_{x,u_k:k\neq i} P(x|u) \prod_{k\neq i} \pi_X^{(t)}(u_k) \\ &= \alpha \int_{x,u_k:k\neq i} P(x, u_k : k \neq i|u_i) \\ &= \alpha\,.\end{aligned}$$

Finally for HLBP, we may be able to calculate some messages exactly. For example, suppose a discrete node has only discrete parents. We can always calculate the messages sent from this node to its neighbors exactly. In this case, we should avoid using Monte Carlo sampling.

## 6 Experimental Results

We tested the HLBP algorithm on two benchmark hybrid Bayesian networks: emission network (Lauritzen, 1992) and its extension augmented emission network (Lerner et al., 2001) shown in Figure 4(a). Note that HLBP is applicable to more general hybrid models; We choose the networks only for comparison purpose. To evaluate how well HLBP performs, we discretized the ranges of continuous variables to 50 intervals and then calculated the Hellinger's distance (Kokolakis and Nanopoulos, 2001) between the results of HLBP and the exact solutions obtained by a massive amount of computation (likelihood weighting with 100M samples) as the error for HLBP. All results reported are the average of 50 runs of the experiments.

### 6.1 Parameter Selection

HLBP has several tunable parameters. We have number of samples for estimating messages (number of message samples), number of samples for the integration (number of integration samples) in Equation 3. The most dramatic influence on precision comes from the number of message samples, shown as in Figure 4(b). Counter intuitively, the number of integration samples does not have as big impact as we might think (see Figure 4(c) with 1K message samples). The reason we believe is that when we draw a lot of samples for messages, the precision of each sample becomes less critical. In our experiments, we set the number of message samples to $1,000$ and the number of integration samples to 12. For the EM algorithm for estimating MGs, we typically set the regularization constant for preventing over fitting to 0.8, stopping likelihood threshold to 0.001, and the number of components in mixtures of Gaussian to 2.

We also compared the performance of two samplers for product of MGs: Gibbs sampler (Sudderth et al., 2003) and the importance sampler in Section 3. As we can see from Figure 4(b,c), when the number of message samples is small, Gibbs sampler has slight advantage over the importance sampler. As the num-

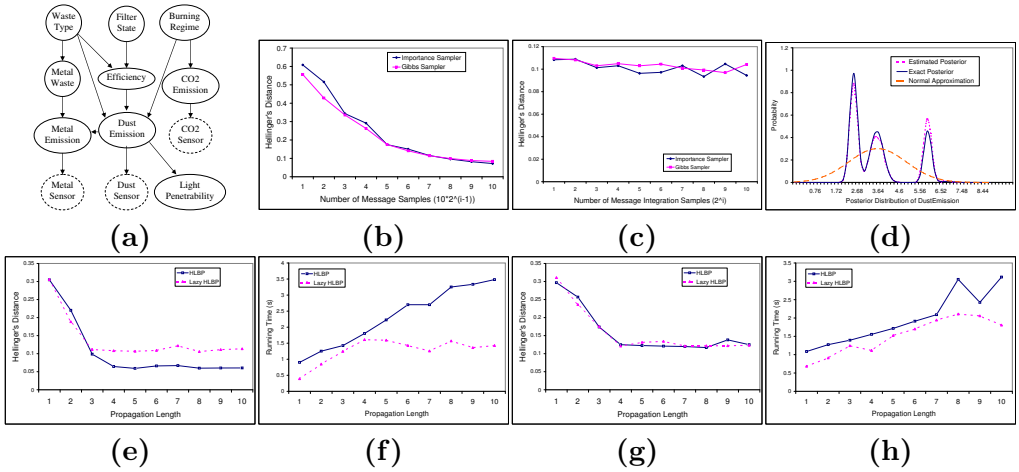



Figure 4: **(a)** Emission network (without dashed nodes) and augmented emission network (with dashed nodes). **(b,c)** The influence of number of message samples and number of message integration samples on the precision of HLBP on augmented Emission network CO2Sensor and DustSensor both observed to be true and Penetrability to be 0.5. **(d)** Posterior probability distribution of DustEmission when observing CO2Emission to be $-1.3$, Penetrability to be 0.5 and WasteType to be 1 in Emission network. **(e,f,g,h)** Results of HLBP and Lazy HLBP: (e) error on emission, (f) running time on emission, (g) error on augmented emission, (h) running time on augmented emission.

ber of message samples increases, the difference becomes negligible. Since the importance sampler is much more efficient, we use it in all our other experiments.

### 6.2 Results on Emission Networks

We note that mean and variance alone provide only limited information about the actual posterior probability distributions. Figure 4(d) shows the posterior probability distribution of node DustEmission when observing CO2Emission at $-1.3$, Penetrability at 0.5, and WasteType at 1. We also plot in the same figure the corresponding normal approximation with mean 3.77 and variance 1.74. We see that the normal approximation does not reflect the true posterior. While the actual posterior distribution has a multimodal shape, the normal approximation does not tell us where the mass really is. We also report the estimated posterior probability distribution of DustEmission by HLBP. HLBP seemed able to estimate the shape of the actual distribution very accurately.

In Figures 4(e,f), we plot the error curves of HLBP and Lazy HLBP (HLBP enhanced by Lazy LBP) as a function of the propagation length. We can see that HLBP needs only several steps to converge. Furthermore, HLBP achieves better precision than its lazy version, but Lazy HLBP is much more efficient than HLBP. Theoretically, Lazy LBP should not affect the results of HLBP but only improve its efficiency if the messages are calcualted exactly. However, we use importance sampling to estimate the messages. Since we use the messages from the last iteration as the importance functions, iterations will help improving the functions.

We also tested the HLBP algorithm on the augmented Emission network (Lerner et al., 2001) with CO2Sensor and DustSensor both observed to be true and Penetrability to be 0.5 and report the results in Figures 4(g,h). We again observed that not many iterations are needed for HLBP to converge. In this case, Lazy HLBP provides comparable results while improving the efficiency of HLBP.

## 7 Conclusion

The contribution of this paper is two-fold. First, we propose the *Hybrid Loopy Belief Propagation* algorithm (HLBP). The algorithm is general enough to deal with general hybrid Bayesian networks that contain mixtures of discrete and continuous variables and may represent linear or nonlinear equations and arbitrary probability distributions and naturally accommodate the scenario where discrete variables have continuous parents. Its another advantage is that it approximates the true posterior distributions. Second, we propose an importance sampler to sample from the product of MGs. Its accuracy is comparable to the Gibbs sampler in (Sudderth et al., 2003) but much more efficient given the same number of samples. We anticipate that, just as LBP, HLBP will work well for many practical models and can serve as a promising approximate method for hybrid Bayesian networks.

## Acknowledgements

This research was supported by the Air Force Office of Scientific Research grants F49620–03–1–0187 and FA9550–06–1–0243 and by Intel Research. We thank anonymous reviewers for several insightful comments that led to improvements in the presentation of the paper. All experimental data have been obtained using SMILE, a Bayesian inference engine developed at the Decision Systems Laboratory and available at `http://genie.sis.pitt.edu/`.

## References

B. R. Cobb and P. P. Shenoy. 2005. Hybrid Bayesian networks with linear deterministic variables. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI–05)*, pages 136–144, AUAI Press Corvallis, Oregon.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society*, B39:1–38.

C. T. Kelley. 2003. *Solving Nonlinear Equations with Newton's Method.*

G. Kokolakis and P.H. Nanopoulos. 2001. Bayesian multivariate micro-aggregation under the Hellinger's distance criterion. *Research in official statistics*, 4(1):117–126.

D. Koller, U. Lerner, and D. Angelov. 1999. A general algorithm for approximate inference and its application to hybrid Bayes nets. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 324–333. Morgan Kaufmann Publishers Inc.

S. L. Lauritzen and D. J. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B (Methodological)*, 50(2):157–224.

S.L. Lauritzen. 1992. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108.

U. Lerner, E. Segal, and D. Koller. 2001. Exact inference in networks with discrete children of continuous parents. In *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 319–328. Morgan Kaufmann Publishers Inc.

S. Moral, R. Rumi, and A. Salmeron. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In *Proceedings of Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-01)*, pages 156–167.

K. Murphy, Y. Weiss, and M. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–99)*, pages 467–475, San Francisco, CA. Morgan Kaufmann Publishers.

K. P. Murphy. 1999. A variational approximation for Bayesian networks with discrete and continuous latent variables. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 457–466. Morgan Kaufmann Publishers Inc.

E. Sudderth, A. Ihler, W. Freeman, and A. Willsky. 2003. Nonparametric belief propagation. In *Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR-03).*

M. Tanner. 1993. *Tools for Statistical Inference: Methods for Exploration of Posterior Distributions and Likelihood Functions.* Springer, New York.

# Probabilistic Independence of Causal Influences

Adam Zagorecki
Engineering Systems Department
Defence Academy of the UK
Cranfield University
Shrivenham, UK

Marek Druzdzel
Decision Systems Laboratory
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15260, USA

**Abstract**

One practical problem with building large scale Bayesian network models is an exponential growth of the number of numerical parameters in conditional probability tables. Obtaining large number of probabilities from domain experts is too expensive and too time demanding in practice. A widely accepted solution to this problem is the assumption of independence of causal influences (ICI) which allows for parametric models that define conditional probability distributions using only a number of parameters that is linear in the number of causes. ICI models, such as the noisy-OR and the noisy-AND gates, have been widely used by practitioners. In this paper we propose PICI, probabilistic ICI, an extension of the ICI assumption that leads to more expressive parametric models. We provide examples of three PICI models and demonstrate how they can cope with a combination of positive and negative influences, something that is hard for noisy-OR and noisy-AND gates.

## 1 INTRODUCTION

Bayesian networks (Pearl, 1988) have proved their value as a modeling tool in many distinct domains. One of the most serious bottlenecks in applying this modeling tool is the costly process of creating the model. Although practically applicable algorithms for learning BN from data are available (Heckerman, 1999), still there are many applications that lack quality data and require using an expert's knowledge to build models.

One of the most serious problems related to building practical models is the large number of conditional probability distributions needed to be specified when a node in the graph has large number of parent nodes. In the case of discrete variables, which we assume in this paper, conditional probabilities are encoded in the form of conditional probability tables (CPTs) that are indexed by all possible combinations of parent states. For example, 15 binary parent variables result in over 32,000 parameters, a number that is impossible to specify in a direct manner. There are two qualitatively different approaches to avoiding the problem of specifying large CPTs. The first is to exploit internal structure within a CPT – which basically means to encode efficiently symmetries in CPTs (Boutilier et al., 1996; Boutilier et al., 1995). The other is to assume some model of interaction among causes (parent influences) that defines the effect's (child node's) CPT. The most popular class of model in this category is based on the concept known as *causal independence* or *independence of causal influences* (ICI) (Heckerman and Breese, 1994) which we describe in Section 2. These two approaches should be viewed as complementary. It is because they are able to capture two distinct types of interactions between causes.

In practical applications, the noisy-OR (Good, 1961; Pearl, 1988) model together with its extension capable of handling multi-valued variables, the noisy-MAX (Henrion, 1989), and the complementary models the noisy-AND/MIN (Díez and Druzdzel, 2002) are the most often applied ICI models. One of the ob-

vious limitations of these models is that they capture only a small set of patterns of interactions among causes, in particular they do not allow for combining both positive and negative influences. In this paper we introduce an extension to the independence of causal influences which allows us to define models that capture both positive and negative influences. We believe that the new models are of practical importance as practitioners with whom we have had contact often express a need for conditional distribution models that allow for a combination of promoting and inhibiting causes.

The problem of insufficient expressive power of the ICI models has been recognized by practitioners and we are aware of at least two attempts to propose models that are based on the ICI idea, but are not strictly ICI models. The first of them is the *recursive noisy-OR* (Lemmer and Gossink, 2004) which allows the specification of interactions among parent causes, but still is limited to only positive influences. The other interesting proposal is the CAST logic (Chang et al., 1994; Rosen and Smith, 1996) which allows for combining both positive and negative influences in a single model, however detaches from a probabilistic interpretation of the parameters, and consequently leads to difficulties with their interpretation and it can not be exploited to speed-up inference.

The remainder of this paper is organized as follows. In Section 2, we briefly describe independence of causal influences. In Section 3 we discuss the amechanistic property of the ICI models, while in Section 4 we introduce an extension of ICI – a new family of models – probabilistic independence of causal influences. In the following two Sections 5 and 6, we introduce two examples of models for local probability distributions, that belong to the new family. Finally, we conclude our paper with discussion of our proposal in Section 7.

## 2 INDEPENDENCE OF CAUSAL INFLUENCES (ICI)

In this section we briefly introduce the concept of independence of causal influences (ICI). First

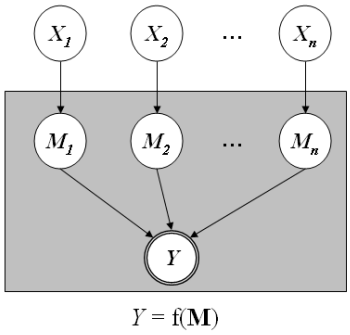



Figure 1: General form of independence of causal interactions

however we introduce notation used throughout the paper. We denote an effect (child) variable as $Y$ and its $n$ parent variables as $\mathbf{X} = \{X_1, \ldots, X_n\}$. We denote the states of a variable with lower case, e.g. $X = x$. When it is unambiguous we use $x$ instead of $X = x$.

In the ICI model, the interaction between variables $X_i$ and $Y$ is defined by means of (1) the *mechanism* variables $M_i$, introduced to quantify the influence of each cause on the effect separately, and (2) the deterministic function $f$ that maps the outputs of $M_i$ into $Y$. Formally, the causal independence model is a model for which two independence assertions hold: (1) for any two mechanism variables $M_i$ and $M_j$ $(i \neq j)$ $M_i$ is independent of $M_j$ given $X_1, \ldots, X_n$, and (2) $M_i$ and any other variable in the network that does not belong to the causal mechanism are independent given $X_1, \ldots, X_n$ and $Y$. An ICI model is shown in Figure 1.

The most popular example of an ICI model is the noisy-OR model. The noisy-OR model assumes that all variables involved in the interaction are binary. The mechanism variables in the context of the noisy-OR are often referred to as *inhibitors*. The inhibitors have the same range as $Y$ and their CPTs are defined as follows:

$$\begin{aligned} P(M_i = y | X_i = x_i) &= p_i \\ P(M_i = y | X_i = \overline{x}_i) &= 0\ . \end{aligned} \qquad (1)$$

Function $f$ that combines the individual influences is the deterministic OR. It is important

to note that the domain of the function defining the individual influences are the outcomes (states) of $Y$ (each mechanism variable maps $Range(X_i)$ to $Range(Y)$). This means that $f$ is of the form $Y = f(M_1, \ldots, M_n)$, where typically all variables $M_i$ and $Y$ take values from the same set. In the case of the noisy-OR model it is $\{y, \overline{y}\}$. The noisy-MAX model is an extension of the noisy-OR model to multi-valued variables where the combination function is the deterministic MAX defined over $Y$'s outcomes.

## 3 AMECHANISTIC PROPERTY

The amechanistic property of the causal interaction models was explicated by Breese and Heckerman, originally under the name of *atemporal* (Heckerman, 1993), although later the authors changed the name to *amechanistic* (Heckerman and Breese, 1996). The amechanistic property of ICI models relates to a major problem of this proposal — namely the problem of determining semantic meaning of causal mechanisms. In practice, it is often impossible to say anything about the nature of causal mechanisms (as they can often be simply artificial constructs for the sake of modeling) and therefore they, or their parameters, can not be specified explicitly. Even though Heckerman and Breese proposed a strict definition of the amechanistic property, in this paper we will broaden this definition and assume that an ICI model is an amechanistic model, when its parameterization can be defined exclusively by means of (a subset) of conditional probabilities $P(Y|\mathbf{X})$ without mentioning $M_i$s explicitly. This removes the burden of defining mechanisms directly.

To achieve this goal it is assumed that one of the states of each cause $X_i$ is a special state (also referred to as as the *distinguished* state). Usually such a state is the normal state, like *ok* in hardware diagnostic systems or *absent* for a disease in a medical system, but such association depends on the modeled domain. We will use * symbol to denote the distinguished state. Given that all causes $X_i$ are in their distinguished states, the effect variable $Y$ is guaranteed to be in its distinguished state. The idea is to allow for easy elicitation of parameters of the intermediate nodes $M_i$, even though these can not be observed directly. This is achieved through a particular way of setting (controlling) the causes $X_i$. Assuming that all causes except for a single cause $X_i$ are in their distinguished states and $X_i$ is in some state (not distinguished), it is easy to determine the probability distribution for the hidden variable $M_i$.

Not surprisingly, the noisy-OR model is an example of an amechanistic model. In this case, the distinguished states are usually *false* or *absent* states, because the effect variable is guaranteed to be in the distinguished state given that all the causes are in their distinguished states $X_i = \overline{x}_i$. Equation 1 reflects the amechanistic assumption and results in the fact that the parameters of the mechanisms (inhibitors) can be obtained directly as the conditional probabilities: $P(Y = y|\overline{x}_1, \ldots, \overline{x}_{i-1}, x_i, \overline{x}_{i+1}, \ldots, \overline{x}_n)$. Similarly, the noisy-MAX is an amechanistic model – it assumes that each parent variable has a distinguished state (arbitrarily selected) and the effect variable has a distinguished state. In the case of the effect variable, the distinguished state is assumed to be the lowest state (with respect to the ordering relation imposed on the effect variable's states). We strongly believe that the amechanistic property is highly significant from the point of view of knowledge acquisition. Even though we introduce a parametric model instead of a traditional CPT, the amechanistic property causes the parametric model to be defined in terms of a conditional probability distribution $P(Y|\mathbf{X})$ and, therefore, is conceptually consistent with the BN framework. We believe that the specification of a parametric model in terms of probabilities has contributed to the great popularity of the noisy-OR model.

## 4 PROBABILISTIC ICI

The combination function in the ICI models is defined as a mapping of mechanisms' states into the states of the effect variable $Y$. Therefore, it can be written as $Y = f(\mathbf{M})$, where $\mathbf{M}$ is a vector of mechanism variables. Let $Q_i$ be a set of parameters of CPT of node $M_i$, and

$\mathbf{Q} = \{Q_1, \ldots, Q_n\}$ be a set of all parameters of all mechanism variables. Now we define the new family *probabilistic independence of causal interactions* (PICI) for local probability distributions. A PICI model for the variable $Y$ consists of (1) a set of $n$ mechanism variables $M_i$, where each variable $M_i$ corresponds to exactly one parent $X_i$ and has the same range as $Y$, and (2) a combination function $f$ that transforms a set of probability distributions $Q_i$ into a single probability distribution over $Y$. The mechanisms $M_i$ in the PICI obey the same independence assumptions as in the ICI. The PICI family is defined in a way similar to the ICI family, with the exception of the combination function, that is defined in the form $P(Y) = f(\mathbf{Q}, \mathbf{M})$. The PICI family includes both ICI models, which can be easily seen from its definition, as $f(\mathbf{M})$ is a subset of $f(\mathbf{Q}, \mathbf{M})$, assuming $\mathbf{Q}$ is the empty set.

In other words, in the case of ICI for a given instantiation of the states of the mechanism variables, the state of $Y$ is a function of the states of the mechanism variables, while for the PICI the distribution over $Y$ is a function of the states of the mechanism variables and some parameters $\mathbf{Q}$.

Heckerman and Breese (Heckerman, 1993) identified other forms (or rather properties) of the ICI models that are interesting from the practical point of view. We would like to note that those forms (decomposable, multiple decomposable, and temporal ICI) are related to properties of the function $f$, and can be applied to the PICI models in the same way as they are applied to the ICI models.

## 5 NOISY-AVERAGE MODEL

In this section, we propose a new local distribution model that is a PICI model. Our goal is to propose a model that (1) is convenient for knowledge elicitation from human experts by providing a clear parameterization, and (2) is able to express interactions that are impossible to capture by other widely used models (like the noisy-MAX model). With this model we are interested in modeling positive and negative influences on the effect variable that has a distinguished state in the middle of the scale.

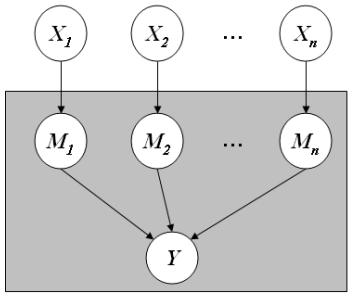



Figure 2: BN model for probabilistic independence of causal interactions, where $P(Y|\mathbf{M}) = f(\mathbf{Q}, \mathbf{M})$.

We assume that the parent nodes $X_i$ are discrete (not necessarily binary, nor an ordering relation between states is required), and each of them has one distinguished state, that we denote as $x_i^*$. The distinguished state is not a property of a parent variable, but rather a part of a definition of a causal interaction model — a variable that is a parent in two causal independence models may have different distinguished states in each of these models. The effect variable $Y$ also has its distinguished state, and by analogy we will denote it by $y^*$. The range of the mechanism variables $M_i$ is the same as the range of $Y$. Unlike the noisy-MAX model, the distinguished state may be in the middle of the scale.

In terms of parameterization of the mechanisms, the only constraint on the distribution of $M_i$ conditional on $X_i = x_i^*$ is:

$$\begin{aligned} P(M_i = m_i^* | X_i = x_i^*) &= 1 \\ P(M_i \neq m_i^* | X_i = x_i^*) &= 0\ , \end{aligned} \qquad (2)$$

while the other parameters in the CPT of $M_i$ can take arbitrary values.

The definition of the CPT for Y is a key element of our proposal. In the ICI models, the CPT for $Y$ was by definition constrained to be a deterministic function, mapping states of $M_i$s to the states of $Y$. In our proposal, we define the CPT of $Y$ to be a function of probabilities

of the $M_i$s:

$$P(y|\mathbf{x}) = \begin{cases} \prod_{i=1}^{n} P(M_i = y^*|x_i) & \text{for } y = y^* \\ \frac{\alpha}{n} \sum_{i=1}^{n} P(M_i = y|x_i) & \text{for } y \neq y^* \end{cases}$$

where $\alpha$ is a normalizing constant discussed later. Let $q_i^{j^*} = q_i^*$. For simplicity of notation assume that $q_i^j = P(M_i = y^j|x_i)$, $q_i^* = P(M_i = y^*|x_i)$, and $D = \prod_{i=1}^{n} P(M_i = y^*|x_i)$. Then we can write:

$$\sum_{j=1}^{m_y} P(y_j|\mathbf{x}) = D + \sum_{j=1, j\neq j^*}^{m_y} \frac{\alpha}{n} \sum_{i=1}^{n} q_i^j =$$

$$= D + \frac{\alpha}{n} \sum_{j=1, j\neq j^*}^{m_y} \sum_{i=1}^{n} q_i^j = D + \frac{\alpha}{n} \sum_{i=1}^{n} (1 - q_i^*),$$

where $m_y$ is the number of states of $Y$. Since the sum on the left must equal 1, as it defines the probability distribution $P(Y|\mathbf{x})$, we can calculate $\alpha$ as:

$$\alpha = \frac{n(1-D)}{\sum_{i=1}^{n} (1 - q_i^*)} \ .$$

The definition above does not define $P(Y|\mathbf{M})$ but rather $P(Y|\mathbf{X})$. It is possible to calculate $P(Y|\mathbf{M})$ from $P(Y|\mathbf{X})$, though it is not needed to use the model. Now we discuss how to obtain the probabilities $P(M_i|X_i)$. Using definition of $P(y|\mathbf{x})$ and the amechanistic property, this task amounts to obtaining the probabilities of $Y$ given that $X_i$ is in its non-distinguished state and all other causes are in their distinguished states (in a very similar way to how the noisy-OR parameters are obtained). $P(y|\mathbf{x})$ in this case takes the form:

$$P(Y = y|x_1^*, \ldots, x_{i-1}^*, x_i, \ldots, x_{i+1}^*, x_n^*) =$$

$$= P(M_i = y|x_i) \ , \qquad (3)$$

and, therefore, defines an easy and intuitive way for parameterizing the model by just asking for conditional probabilities, in a very similar way to the noisy-OR model. It is easy to notice that $P(Y = y^*|x_1^*, \ldots, x_i^*, \ldots, x_n^*) = 1$, which poses a constraint that may be unacceptable from a modeling point of view. We can address this limitation in a very similar way to the noisy-OR model, by assuming a dummy variable $X_0$ (often referred to as *leak*), that stands for all unmodeled causes and is assumed to be always in some state $x_0$. The leak probabilities are obtained using:

$$P(Y = y|x_1^*, \ldots, x_n^*) = P(M_0 = y) \ .$$

However, this slightly complicates the schema for obtaining parameters $P(M_i = y|x_i)$. In the case of the leaky model, the equality in Equation 3 does not hold, since $X_0$ acts as a regular parent variable that is in a non-distinguished state. Therefore, the parameters for other mechanism variables should be obtained using conditional probabilities $P(Y = y|x_1^*, \ldots, x_i, \ldots, x_n^*)$, $P(M_0 = y|x_0)$ and the combination function. This implies that the acquired probabilities should fulfil some non-trivial constraints. Because of space limitation, we decided to skip the discussion of these constraints. In a nutshell, these constraints are similar in nature to constraints for the leaky noisy-MAX model. These constraints should not be a problem in practice, when $P(M_0 = y^*)$ is large (which implies that the leak cause has marginal influence on non-distinguished states).

Now we introduce an example of the application of the new model. Imagine a simple diagnostic model for an engine cooling system. The pressure sensor reading (S) can be in three states high, normal, or low {*hi, nr, lo*}, that correspond to pressure in a hose. Two possible faults included in our model are: *pump failure* (P) and *crack* (C). The pump can malfunction in two distinct ways: work non-stop instead of adjusting its speed, or simply fail and not work at all. The states for *pump failure* are: {*ns, fa, ok*}. For simplicity we assume that the crack on the hose can be *present* or *absent* {*pr,ab*}. The BN for this problem is presented in Figure 3. The noisy-MAX model is not appropriate here, because the distinguished state of the effect variable (S) does not correspond to the lowest value in the ordering relation. In other words, the neutral value is not one of the extremes, but lies in the middle, which makes use of the MAX function over the states inappropriate. To apply the noisy-average model, first we should identify the distinguished states of the variables.

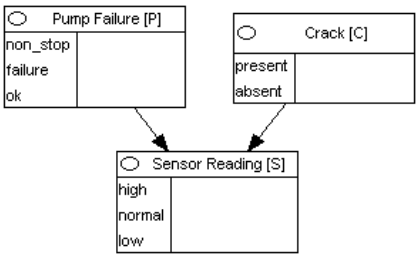



Figure 3: BN model for the pump example.

In our example, they will be: *normal* for *sensor reading*, *ok* for *pump failure* and *absent* for *crack*. The next step is to decide whether we should add an influence of non-modeled causes on the sensor (a leak probability). If such an influence is not included, this would imply that $P(S = nr * | P = ok*, C = ab*) = 1$, otherwise this probability distribution can take arbitrary values from the range $(0, 1]$, but in practice it should always be close to 1.

Assuming that the influence of non-modeled causes is not included, the acquisition of the mechanism parameters is performed directly by asking for conditional probabilities of form $P(Y|x_1^*, \ldots, x_i, \ldots, x_n^*)$. In that case, a typical question asked of an expert would be: *What is the probability of the sensor being in the low state, given that a crack was observed but the pump is in state ok?* However, if the unmodeled influences were significant, an adjustment for the leak probability is needed. Having obtained all the mechanism parameters, the noisy-average model specifies a conditional probability in a CPT by means of the combination function.

Intuitively, the noisy-average combines the various influences by averaging probabilities. In case where all active influences (the parents in non-distinguished states) imply high probability of one value, this value will have a high posterior probability, and the synergetic effect will take place similarly to the noisy-OR/MAX models. If the active parents will 'vote' for different effect's states, the combined effect will be an average of the individual influences. Moreover, the noisy-average model is a decomposable model — the CPT of $Y$ can be decomposed in pairwise relations (Figure 4) and such a decomposition can be exploited for sake of improved inference speed in the same way as for decomposable ICI models.

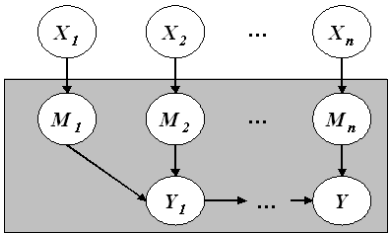



Figure 4: Decomposition of a combination function.

It is important to note that the noisy-average model does not take into account the ordering of states (only the distinguished state is treated in a special manner). If a two causes have high probability of *high* and *low* pressure, one should not expect that combined effect will have probability of normal state (the value in-between).

## 6 AVERAGE MODEL

Another example of a PICI model we want to present is the model that averages influences of mechanisms. This model highlights another property of the PICI models that is important in practice. If we look at the representation of a PICI model, we will see that the size of the CPT of node $Y$ is exponential in the number of mechanisms (or causes). Hence, in general case it does not guarantee a low number of distributions. One solution is to define a combination function that can be expressed **explicitly** in the form of a BN but in such a way that it has significantly fewer parameters. In the case of ICI models, the decomposability property (Heckerman and Breese, 1996) served this purpose, and can do too for in PICI models. This property allows for significant speed-ups in inference.

In the average model, the probability distribution over $Y$ given the mechanisms is basically a ratio of the number of mechanisms that are in given state divided by the total number of mechanisms (by definition $Y$ and $\mathbf{M}$ have the

same range):

$$P(Y = y|M_1, \ldots, M_n) = \frac{1}{n} \sum_{i=1}^{n} I(M_i = y) \ . \tag{4}$$

where $I$ is the identity function. Basically, this combination function says that the probability of the effect being in state $y$ is the ratio of mechanisms that result in state $y$ to all mechanisms. Please note that the definition of how a cause $X_i$ results in the effect is defined in the probability distribution $P(M_i|X_i)$. The pairwise decomposition can be done as follows:

$$P(Y_i = y|Y_{i-1} = a, M_n = b)$$

$$= \frac{i}{i+1} I(y = a) + \frac{1}{i+1} I(y = b) \ ,$$

for $Y_2, \ldots, Y_n$. $Y_1$ is defined as:

$$P(Y_1 = y|M_1 = a, M_2 = b) =$$

$$= \frac{1}{2} I(y = a) + \frac{1}{2} I(y = b) \ .$$

The fact that the combination function is decomposable may be easily exploited by inference algorithms. Additionally, we showed that this model presents benefits for learning from small data sets (Zagorecki et al., 2006).

Theoretically, this model is amechanistic, because it is possible to obtain parameters of this model (probability distributions over mechanism variables) by asking an expert only for probabilities in the form of $P(Y|\mathbf{X})$. For example, assuming variables in the model are binary, we have $2n$ parameters in the model. It would be enough to select $2n$ arbitrary probabilities $P(Y|\mathbf{X})$ out of $2^n$ and create a set of $2n$ linear equations applying Equation 4. Though in practice, one needs to ensure that the set of equations has exactly one solution what in general case is not guaranteed.

As an example, let us assume that we want to model classification of a threat at a military checkpoint. There is an expected terrorist threat at that location and there are particular elements of behavior that can help spot a terrorist. We can expect that a terrorist can approach the checkpoint in a large vehicle, being the only person in the vehicle, try to carry the attack at rush hours or time when the security is less strict, etc. Each of these behaviors is not necessarily a strong indicator of terrorist activity, but several of them occurring at the same time may indicate possible threat.

The average model can be used to model this situation as follows: separately for each of suspicious activities (causes) a probability distribution of terrorist presence given this activity can be obtained which basically means specification of probability distribution of mechanisms. Then combination function for the average model acts as "popular voting" to determine $P(Y|\mathbf{X})$.

The average model draws ideas from the linear models, but unlike the linear models, the linear sum is done over probabilities (as it is PICI), and it has explicit hidden mechanism variables that define influence of single cause on the effect.

## 7 CONCLUSIONS

In this paper, we formally introduced a new class of models for local probability distributions that is called PICI. The new class is an extension of the widely accepted concept of independence of causal influences. The basic idea is to relax the assumption that the combination function should be deterministic. We claim that such an assumption is not necessary either for clarity of the models and their parameters, nor for other aspects such as convenient decompositions of the combination function that can be exploited by inference algorithms.

To support our claim, we presented two conceptually distinct models for local probability distributions that address different limitations of existing models based on the ICI. These models have clear parameterizations that facilitate their use by human experts. The proposed models can be directly exploited by inference algorithms due to fact that they can be explicitly represented by means of a BN, and their combination function can be decomposed into a chain of binary relationships. This property has been recognized to provide significant inference speed-ups for the ICI models. Finally, because

they can be represented in form of hidden variables, their parameters can be learned using the EM algorithm.

We believe that the concept of PICI may lead to new models not described here. One remark we shall make here: it is important that new models should be explicitly expressible in terms of a BN. If a model does not allow for compact representation and needs to be specified as a CPT for inference purposes, it undermines a significant benefit of models for local probability distributions – a way to avoid using large conditional probability tables.

**Acknowledgements**

This research was supported by the Air Force Office of Scientific Research grants F49620–03–1–0187 and FA9550–06–1–0243 and by Intel Research. While we take full responsibility for any errors in the paper, we would like to thank anonymous reviewers for several insightful comments that led to improvements in the presentation of the paper, Tsai-Ching Lu for inspiration, invaluable discussions and critical comments, and Ken McNaught for useful comments on the draft.

## References

C. Boutilier, R. Dearden, and M. Goldszmidt. 1995. Exploiting structure in policy construction. In Chris Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1104–1111, San Francisco. Morgan Kaufmann.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. 1996. Context-specific independence in Bayesian networks. In *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 115–123, San Francisco, CA. Morgan Kaufmann Publishers.

K.C. Chang, P.E. Lehner, A.H. Levis, Abbas K. Zaidi, and X. Zhao. 1994. On causal influence logic. *Technical Report for Subcontract no. 26-940079-80.*

F. J. Díez and Marek J. Druzdzel. 2002. Canonical probabilistic models for knowledge engineering. Forthcoming.

I. Good. 1961. A causal calculus (I). *British Journal of Philosophy of Science*, 11:305–318.

D. Heckerman and J. Breese. 1994. A new look at causal independence. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)*, pages 286–292, San Francisco, CA. Morgan Kaufmann Publishers.

D. Heckerman and J. Breese. 1996. Causal independence for probability assessment and inference using Bayesian networks. In *IEEE, Systems, Man, and Cybernetics*, pages 26:826–831.

D. Heckerman. 1993. Causal independence for knowledge acquisition and inference. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (UAI–93)*, pages 122–127, Washington, DC. Morgan Kaufmann Publishers.

D. Heckerman. 1999. A tutorial on learning with Bayesian networks. In *Learning in Graphical Models*, pages 301–354, Cambridge, MA, USA. MIT Press.

M. Henrion. 1989. Some practical issues in constructing belief networks. In L.N. Kanal, T.S. Levitt, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pages 161–173. Elsevier Science Publishing Company, Inc., New York, N. Y.

J. F. Lemmer and D. E. Gossink. 2004. Recursive noisy-OR: A rule for estimating complex probabilistic causal interactions. *IEEE Transactions on Systems, Man and Cybernetics*, (34(6)):2252 – 2261.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., San Mateo, CA.

J. A. Rosen and W. L. Smith. 1996. Influence net modeling and causal strengths: An evolutionary approach. In *Command and Control Research and Technology Symposium.*

A. Zagorecki, M. Voortman, and M. Druzdzel. 2006. Decomposing local probability distributions in Bayesian networks for improved inference and parameter learning. In *FLAIRS Conference.*

# Author Index

| | |
|---|---|
| Title: | Proceedings of the Third European Workshop<br>on Probabilitic Graphical Models |
| Editors: | Milan Studený and Jiří Vomlel |
| Published by: | Zeithamlová Milena, Ing. - Agentura Action M<br>Vršovická 68, 101 00 Praha 10, Czech Republic<br>http://www.action-m.com |
| Number of Pages: | 344 |
| Number of Copies: | 80 |
| Year of Publication: | 2006 |
| Place of Publication: | Prague |
| First Edition | |
| Printed by: | Reprostředisko UK MFF<br>Charles University in Prague - Faculty of Mathematics and Physics<br>Sokolovská 83, 186 75 Praha 8 - Karlín, Czech Republic |

ISBN 80-86742-14-8