Proceedings of the
4th European Workshop on Probabilistic Graphical Models

Proceedings of the

4th European Workshop on Probabilistic Graphical Models

Hirtshals, Denmark, September 17-19, 2008

Manfred Jaeger, Thomas D. Nielsen, Editors

Sponsored by

HUGIN EXPERT

Cover design by M. Jaeger and T. D. Nielsen.
Electronic version of 'Dania Regnum' map created by the UCLA
library and used with permission.

Conference Organization

Organizing Committee

Finn V. Jensen
Uffe Kjærulff

Program Chairs

Manfred Jaeger
Thomas D. Nielsen

Program Committee

Concha Bielza
Adnan Darwiche
Luis M. de Campos
Francisco J. Díez
Marek J. Druzdzel
Ad Feelders
Julia Flores
José A. Gámez
Christophe Gonzales
Rolf Haenni
Juan F. Huete
David Rios Insua
Radim Jiroušek
Kristian Kersting
Tomáš Kočka
Helge Langseth
Pedro Larrañaga
Peter Lucas
Fero Matúš
Serafín Moral
Jens D. Nielsen
Kristian G. Olesen
José M. Peña
José M. Puerta
Silja Renooij
Teemu Roos
Antonio Salmerón
Prakash P. Shenoy
James Q. Smith
Harald Steck
Milan Studený
Marco Valtorta
Linda van der Gaag
Jirka Vomlel
Marta Vomlelová
Jon Williamson
Pierre-Henri Wuillemin
Yang Xiang
Marco Zaffalon
Nevin Zhang

Additional reviewers

Mark Chavira
Tao Chen
Arthur Choi
Juan L. Mateo
James Park
Jingsong Wang

Preface

The European Workshop on Probabilistic Graphical Models (PGM) is a biennial workshop, which was first held in Cuenca, Spain, in 2002. The two following workshops took place in Leiden, the Netherlands (2004) and in Prague, the Czech Republic (2006). This year's workshop will take place in Hirtshals, Denmark, from September 17th to September 19th, 2008. Hirtshals is located on the north-western coast of Denmark, approximately 70 kilometers from Aalborg.

The aim of the workshop is to bring together people interested in probabilistic graphical models and provide a forum for discussion of the latest research developments in this field. The workshop is organized so as to facilitate discussions and collaboration among the participants also outside the workshop sessions.

This year there were 47 papers submitted to the workshop. The papers went though a rigorous reviewing process, where the majority of the papers were reviewed by three program committee members. Of the 47 submitted papers, 20 papers were accepted for plenary presentation and 19 papers were accepted for poster presentation.

The reviewing process took place in the period from May 9th to June 10th. There were 40 members in the program committee, and each committee member received approximately 3.525 papers for review. In addition to the program committee members, there were also six additional reviewers to help with the reviewing process.

We would like to thank the program committee members and the reviewers for all the work they have done in order to make PGM'08 a success. The reviews that we received were of general high quality and included constructive comments to help the authors improve upon their papers.

Finally, we would like to thank Helene Blaschke, Lene Winter Even, and Ulla Øland for their help organizing the conference.

September 2008 Manfred Jaeger and Thomas D. Nielsen

Table of Contents

Query-Based Diagnostics 1
John M. Agosta, Thomas R. Gardos and Marek J. Druzdzel

High-Dimensional Probability Density Estimation with Randomized Ensembles of Tree Structured Bayesian Networks 9
Sourour Ammar, Philippe Leray, Boris Defourny and Louis Wehenkel

Generalized Loopy 2U: A New Algorithm for Approximate Inference in Credal Networks 17
Alessandro Antonucci, Marco Zaffalon, Yi Sun and Cassio P. de Campos

Carmen: An Open Source Project for Probabilistic Graphical Models 25
Manuel Arias and Francisco J. Díez

Bayesian Networks: the Parental Synergy 33
Janneke H. Bolt

A Score Based Ranking of the Edges for the PC Algorithm 41
A. Cano, M. Gómez-Olmedo and S. Moral

A Bayesian Approach to Estimate Probabilities in Classification Trees 49
Andrés Cano, Andrés Masegosa and Serafín Moral

Efficient Model Evaluation in the Search-Based Approach to Latent Structure Discovery 57
Tao Chen, Nevin L. Zhang and Yi Wang

Measuring Efficiency in Influence Diagram Models 65
Barry R. Cobb

Attribute Clustering Based on Heuristic Tree Partition 73
Jorge Cordero H. and Yifeng Zeng

A Novel Scalable and Correct Markov Boundary Learning Algorithm Under Faithfulness Condition 81
Sergio Rodrigues de Morais and Alex Aussem

Marginals of DAG-Isomorphic Independence Models 89
Peter R. de Waal

Policy Explanation in Factored Markov Decision Processes 97
Francisco Elizalde, L. Enrique Sucar, Manuel Luque, Francisco Javier Díez and Alberto Reyes

Learning Naive Bayes Regression Models from Missing Data Using Mixtures of Truncated Exponentials 105
Antonio Fernández, Jens D. Nielsen and Antonio Salmerón

The Probabilistic Interpretation of Model-Based Diagnosis 113
Ildikó Flesch and Peter J. F. Lucas

Efficient Bayesian Network Learning Using EM or Pairwise Deletion 121
Olivier C. H. François

Robust Classification Using Mixtures of Dependency Networks 129
José A. Gámez, Juan L. Mateo, Thomas D. Nielsen and José M. Puerta

Towards Consistency in General Dependency Networks 137
José A. Gámez, Juan L. Mateo and José M. Puerta

Sensitivity of Gaussian Bayesian Networks to Inaccuracies in Their Parameters 145
Miguel A. Gómez-Villegas, Paloma Main and Rosario Susi

Approximate Representation of Optimal Strategies from Influence Diagrams 153
Finn Verner Jensen

Complexity Results for Enumerating MPE and Partial MAP 161
Johan Kwisthout

Parameter Estimation in Mixtures of Truncated Exponentials 169
Helge Langseth, Thomas D. Nielsen, Rafael Rumí and Antonio Salmerón

An Anytime Algorithm for Evaluating Unconstrained Influence Diagrams 177
Manuel Luque, Thomas D. Nielsen and Finn V. Jensen

An Independence of Causal Interactions Model for Opposing Influences 185
Paul P. Maaskant and Marek J. Druzdzel

New Methods for Marginalization in Lazy Propagation 193
Anders L. Madsen

Solving CLQG Influence Diagrams Using Arc-Reversal Operations in a Strong Junction Tree 201
Anders L. Madsen

Computing the Multinomial Stochastic Complexity in Sub-Linear Time 209
Tommi Mononen and Petri Myllymäki

Structural-EM for Learning PDG Models from Incomplete Data 217
Jens D. Nielsen, Rafael Rumí and Antonio Salmerón

Logical Properties of Stable Conditional Independence 225
Mathias Niepert and Dirk Van Gucht

A* Wars: The Fight for Improving A* Search for Troubleshooting with Dependent Actions . 233
Thorsten J. Ottosen and Finn V. Jensen

Discrimination and its Sensitivity in Probabilistic Networks 241
Silja Renooij and Linda C. van der Gaag

An Empirical Analysis of Loopy Belief Propagation in Three Topologies: Grids, Small-World Networks and Random Graphs 249
R. Santana, A. Mendiburu and J. A. Lozano

Factorized Normalized Maximum Likelihood Criterion for Learning Bayesian Network Structures 257
Tomi Silander, Teemu Roos, Petri Kontkanen and Petri Myllymäki

Large Incomplete Sample Robustness in Bayesian Networks 265
Jim Q. Smith and Alireza Daneshkhah

Eliciting Expert Beliefs on the Structure of a Bayesian Network 273
Federico M. Stefanini

A Geometric Approach to Learning BN Structures 281
M. Studený and J. Vomlel

An Influence Diagram Framework for Acting Under Influence by Agents with Unknown Goals 289
Nicolaj Søndberg-Jeppesen and Finn Verner Jensen

Arithmetic Circuits of the Noisy-Or Models 297
Jiří Vomlel and Petr Savicky

Tightly and Loosely Coupled Decision Paradigms in Multiagent Expedition 305
Yang Xiang and Franklin Hanshar

Query-based Diagnostics

John M. Agosta and Thomas R. Gardos
Intel Corporation
2200 Mission College Boulevard
Santa Clara, CA 95054, USA

Marek J. Druzdzel
Faculty of Computer Science, Białystok Technical
University, Wiejska 45A, 15-351 Białystok, Poland
and Decision Systems Laboratory
School of Information Sciences
and Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260, USA

Abstract

We describe an approach to modeling diagnostic problems that is based on a passive observation of a diagnostician's work-flow and recording their findings and final diagnosis, from which the model can be modified directly, or improved by learning from cases so acquired. While the probabilistic model of a system under diagnosis is necessarily simplified, based on three-layer Bayesian networks with canonical interactions among the network variables, we are able to reduce greatly the most important bottleneck—the knowledge engineering effort that goes into model building. Our initial experience with an implementation of this idea suggests that the sacrifices in diagnostic quality are not large, while gains are tremendous.

1 Introduction

It is common knowledge that in today's stage of development of normative methods, i.e., methods based on probability theory and decision theory, it is not the speed of the algorithms that is a bottleneck. Even though probabilistic inference has been shown to be NP-hard (Cooper, 1990) and there exist models that will be too challenging even for a supercomputer, practical systems are capable of solving models consisting of hundreds or even thousands of variables within seconds. The true bottleneck in this approach is the knowledge engineering effort that goes into constructing models. It is our experience, in the domain of machine diagnosis at Intel and elsewhere, that diagnosing a complex manufacturing device requires an initial investment of possibly person-years of knowledge engineering effort. Given that building probabilistic models is a task that requires considerable knowledge and experience, the initial costs are too prohibitive in most practical environments. In order to disseminate the otherwise attractive probabilistic approach, one needs to find ways of overcoming the knowledge engineering bottleneck.

The conventional procedure for implementing knowledge-based systems separates model-building from model usage. Different actors perform tasks of building and running the model at different times (Schreiber et al., 2000). Conventionally a *diagnostic session* consists of a series of observations, each, in turn, generating a query to a previously formulated model that returns recommendations on the next step in the diagnosis. (For ease of exposition, we will

abbreviate the phase "during the steps in a diagnostic session" as simply "during diagnosis.") As the diagnosis progresses, possible causes that initiated the session are clarified, ending in a recommendation of a repair or, in clinical cases, of a treatment and prognosis.

In this paper, we consider an approach that relaxes this constraint by interleaving model building and refinement, and execution. A similar situation occurs with models derived from data: Learning a model has been traditionally a separate activity, often quite computationally intensive, that precedes use of a model for inference. However, recent interest in *online-learning* augments this approach, so that learning and inference are inter-twined (Blum, 1998). *Query-based* diagnosis is analogous in that the model may change during use. However our approach is quite different, and we do not draw upon techniques developed for on-line learning. We explore here two approaches to mingling model refinement with the diagnostic session, one derived from study of work-flow, the other of learning from cases.

By *query-based diagnosis* we mean that the diagnostic model is created and modified in the course of the diagnosis. Is is fluid and not entirely determined before the diagnosis begins. The model applied during the diagnostic session is conditioned on the initial query (sometimes called the *primary-complaint*) and the queries made during the session. It is the structure of the model that changes, not only the state of information consisting of the observations used for inference. The implication of the term is also that the case data on which the model is based may be retrieved after the session begins, and learning from the case data may take place simultaneously with the diagnosis. This is made possible, in part, by the computational power of the current hardware.

In this paper, which is the first time this idea has been applied to simplify and improve elicitation, we first review a few pre-cursors in the literature, explain the foundations of our approach, discuss the system's architecture as it evolved from study of work-flow, give some initial insights on how the computational problems can be addressed, and finally present the first prototype, which can be accessed by the reader on-line.

2 Background

The possibility of learning a model for inference, conditional on specific circumstances arose as advances in algorithms and hardware (e.g. "Moore's law") made real-time learning practical. (Wellman et al., 1992) was an early proponent of building a query-specific probabilistic decision model from a knowledge base. This line of research has matured and proliferated numerous automated Bayes network construction methods, for instance, MEBN that constructs networks from "fragments" based on attributes of the problem. These methods are insightful on translating a database representation into a Bayes network, but do not address the question of elicitation during diagnosis, but raise a different set of elicitation questions.(Laskey, 2008)

An example of a full-fledged system that demonstrates query-driven model-building is the PRIORITIES system, and the COORDINATE system that grew out of it. They implement inference models to predict time intervals of individual's activities by retrieving relevant cases from a database of activities. As the authors describe it:

> Rather than attempting to build a massive static predictive model for all possible queries, we instead focus the analysis by constructing a set of cases from the event database that is consistent with the query at hand.(Horvitz et al., 2002)

Conceivably the range of models that the system is capable of generating makes preconstructing the models practically impossible.

Similarly, Visweswaran and Cooper have proposed supervised classifier learning, under the term "instance-based" classifiers, that generalizes "lazy" learning of classifiers. (Visweswaran and Cooper, 2004) Their approach averages over related instance models to avoid pitfalls of relying on too restrictive a query.

All these approaches condition the model on the query addressed to it; in contrast, we attempt in addition to learn from user's actions while the user is applying the model.

3 Origin of our Approach

As knowledge-engineering practitioners are well aware, elicitation of causal relationships and model structure places large cognitive demands on domain experts. Part of the challenge is the accessibility of knowledge; during an elicitation session, the domain expert is faced with a set of hypothetical circumstances, that often consist of questions about rare occurrences (since descriptions of common occurrences can often be derived from data). One would like to elicit this knowledge when it is current with the activities of the expert. Why not gather this knowledge at the time it is used? During diagnosis the expert is immersed in the problem and the causal relationships have a cognitive immediacy not available generally. Incidentally this may be true at other times, like at the time that the target system is designed and validated.

In our experience, we find that combining model building with receiving recommendations from the model is natural for our users. This became apparent during an application project-review with our client, who had designed an alternate approach for supporting diagnosis. Our application, in conventional fashion, ran inference cycles, based on the user's current observations, to compute posterior probabilities of the faults present in the model. Our client's application closely followed the steps outlined in the work flow for the task. It had options for users to create new observations and to suggest possible causes relevant to the problem during the diagnostic session. It lacked any capability for inference, however, and merely collected cases that could be reviewed in subsequent situations. The challenge was put to us to combine inference with the ability to modify the diagnostic model as evinced by the user's suggestions of unmodeled causal relationships. The manufacturing equipment domain to which this is applied is dynamic with method and process modifications occurring monthly, and without software support that can keep pace with the changes. And it is fair to say that our client has a sophisticated user in mind, whose understanding of the diagnostic problem at hand is on a par with the model they are running. We embraced their approach as one solution to the well-recognized "knowledge elicitation bottleneck."

4 Diagnostic Work-flow

Diagnostic work-flow is the choice of steps available to the user to pursue the diagnosis. The software design codifies the process both to guide the user, and to capture cases for solving similar problems should they arise again. In addition to the existing work-flow we add a mechanism for elicitation of causes in the midst of the problem-solving work-flow that they are familiar with.

The process consists of an interaction that forms a dialog where at each step the application suggests the next actions by ranking possible causes by their posteriors and tests by their diagnostic value, based on observations already entered into the model. This fault set is displayed as a ranked list of possible diagnoses along with an indication of the posterior probabilities of the faults, as computed by the underlying model. In addition a list of suggested observations is displayed from those observations that reside in the model. In suggesting observations, the Bayesian network engine ranks them by expected gain in entropy among the faults.

The user has four choices at each step:

1. Select an existing observation variable to instantiate;
2. Create new test observation;
3. Create a new hypothesized fault; or
4. End the diagnostic cycle by selecting a cause and making the associated repair.

Creating new observations or faults modify the causal structure of the model, requiring the user to indicate what observations a fault manifests, and correspondingly, what faults an observation implies. Causal relationships are the

primary semantics of the model and the user deserves some guidance on when they are necessary. For this we bring up a "pop-up" editor, asking the user to associate a new fault with observations and vice versa. A screen shot of this editor is shown in Figure 1.

A primary intent of building the application around work-flow is to hide the Bayes network from the user. In doing so we make several simplifying assumptions on model structure: The model is a bi-partite graph with occasional context nodes to condition faults; all nodes have binary state spaces; all observation nodes assume Independence of Causal Influences (ICI); and all nodes when initially created are assigned default probabilities so that the user does not need to enter numeric values. These assumptions may appear restrictive, however they should be considered in the context of the feeble guidance that current support tools offer.

By assuming Independence of Causal Influences for all observation nodes, the addition or removal of a cause translates directly into the addition or removal of one arc and modular change in the CPT of the child node. This is critically necessary for modification of the model to incorporate causal knowledge elicited during diagnosis, and makes model modification straightforward. The notion of Independence of Causal Influences (ICI) has been described many times in the literature, usually under a somewhat confusing term *causal independence*. Please see (Díez and Druzdzel, 2008) for a review of canonical models, including ICI models.

4.1 Causal Work-flow Example

Aside from the modeling and computational challenges to updating the model within a diagnostic session, the work-flow of such a session presents no surprises to the user. Consider this "use-case" example of diagnostic support for a simplified automobile diagnosis.

The user begins the session by entering the primary complaint, of *smoke visible in the exhaust.* The application responds by referring to the appropriate, if incomplete, model, and suggests a list of possible causes, such as *worn piston rings*, etc. To isolate the likely cause, the user proposes to do a *compression test*, a test that is not in the existing model. When the user enters the description of the test, she also designates other possible causes which are relevant to it, both from a list of those already in the model, and those she might create, in addition to *worn piston rings*, such as an *exhaust valve leak*. More significant is that she is steered to not designating causes to which the test is not relevant, making the test a good differentiator among them. Similarly the user could add a new possible cause of the primary complaint. She would then be prompted at that step in the session to designate the relevant observations. The session continues through diagnostic cycles by the user taking one of the four actions listed above.

The result of this interaction style is a model that makes tradeoffs that lean toward a more dynamic, up-to-date model at the cost of current model accuracy and completeness. The appropriate degree to which to shift the interaction style toward dynamicism depends strongly on the domain.

5 A Working Prototype

We have implemented the ideas presented in this paper in a prototype diagnostic system, called MARILYN, available to interested readers at the Decision Systems Laboratory's web site at the following location: `http://barcelona.exp.sis.pitt.edu/`. A sister implementation to MARILYN is in preparation at Intel, to be evaluated on machinery maintenance tasks. We will describe the academic prototype below, understanding that the two implementations share many features.

From the point of view of user work-flow, MARILYN appears as a smart data-entry module that collects observables, such as symptoms and test results, context information, such as the age of a device, prior problems with it, etc., and finally, the final diagnosis. MARILYN involves its user in a dialog that focuses on entering a diagnostic case and does not reveal the underlying probabilistic model. The target users are diagnosticians, possibly working in a machine

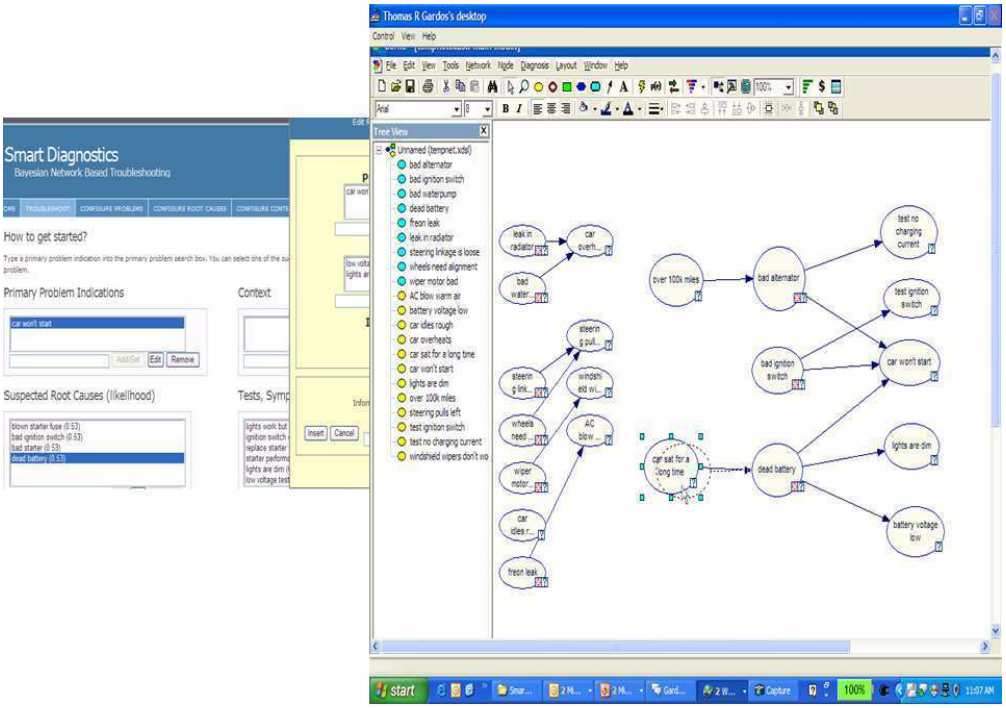


Figure 1: When the user enters a variable name that is not already in the current model, they are presented with a pop-up editor (here shown in yellow and largely hidden from view), for entering causal information. We see here a screen shot of the *GeNIe* editor that shows the model fragments generated by previous diagnoses super-imposed over the panes of the web application. The model network fragments are *not* shown to the user in the application.

repair shop. MARILYN takes a very gentle attitude towards the user—it merely suggests a set of possible diagnoses from among those diagnoses that have been entered in the past that are contained in the current model and implied by the observations entered.

5.1 The System Architecture

The intent of making MARILYN a web-based program is to make the system accessible from any computer through the Internet. MARILYN prototype consists of three elements: (1) user interface running on the client, (2) database of cases, and (3) a Bayesian reasoning engine based on SMILE, both running at the server. Information that is entered by the user is stored in a database (Figure 2 shows the database schema, which illustrates the components of the information collected from the user during a diagnostic session). The database is used by the user interface module in its auto-complete function. The database is also accessed directly by the

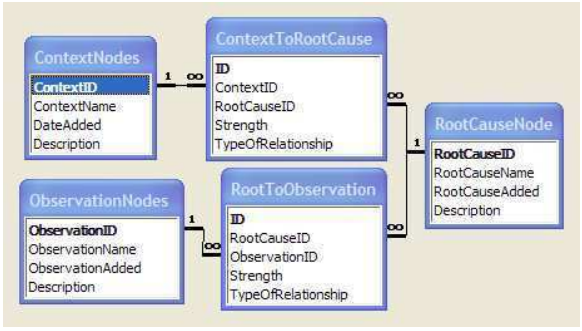


Figure 2: Entity-relationship diagram for the database storing diagnostic information in MARILYN

Bayesian reasoning engine, which constructs on the fly a three-layer Bayesian network that is used in deriving suggested diagnoses and suggested observations. As revealed in Figure 1, it is possible to export the underlying network to GeNIe for the purpose of the examination by a knowledge engineer.

5.2 The Quality of the System's Advice

In our experience, the quality of Marilyn's initial advice (during the first 10-20 diagnostic cases) is low, as the system is not aware of most of the possible diagnoses. As time progresses and the number of cases entered increases, the quality of the model increases and so does the quality of the system's advice. It is our expectation that in such environments as help desks, where tens of hundreds of cases are entered daily, Marilyn's approach will outperform novice diagnosticians fairly quickly.

5.3 Managing Variables

A downside of the system's flexibility toward the users being able to enter any text as the name of variable is the possible proliferation of similar variable names. While its user is typing a name, Marilyn offers suggestions from the database, based on simple techniques for string matching (along the lines of the "auto-complete" function of web browsers). We believe that for a system like Marilyn to truly scale, this flexibility is necessary; however this invites confusion due to model variables that are named slightly differently but stand for the same concept.

In the future we hope to augment the string matching auto-complete feature of Marilyn's user interface with more sophisticated statistical language processing techniques. The intent would be to accommodate common typographic errors using string edit distance measures as well as more challenging synonymous labels through semantic similarity techniques. The ultimate goal is to make this system scale to large models with as little expert user intervention as possible.

Also to support scaling, we believe that it is necessary to equip the system with what we call *expert mode* in which an expert knowledge engineer downloads the underlying Bayesian network, analyzes it, corrects it as necessary, and uploads it back to the system. The frequency of this knowledge base maintenance task depends on the speed of growth of the model. Similarly, the expert might review cases generated by diagnostic sessions that will be used in learning to refine the model. Over time the frequency of reviewing the expert model should converge to a low level.

An issue related to the expert mode and to the initial weak performance of the system is the possibility jump-starting the initial domain model.[1] The quality of this initial model is not critical and the expert can spend just a little bit of time entering the most important domain variables. We expect that the quality of this initial model will be improved fairly quickly as the system is fielded and diagnostic cases entered.

5.4 Managing Model Growth

One technical challenge that we are facing with a system of this type is a possible uncontrolled growth of the underlying model. Model size is normally not a problem, as a reasoning engine such as SMILE is able to handle huge networks and propagate evidence in them within a fraction of a second. What is more troublesome are certain undesirable trends in the network topology. If the number of parents of a single node, for example, grows above 15, this node may pose a considerable challenge to the algorithms. Even though Marilyn represents all conditional probability distributions as *DeMorgan* gates, a variety of ICI model (a detailed exposition of the DeMorgan gate, a natural and intuitive combination of Noisy-OR and Noisy-AND gates can be found as a separate paper in this volume (Maaskant and Druzdzel, 2008)), the size of the conditional probability table is a challenge for the algorithms. Our initial way of dealing with this problem is imposing a maximum on the number of parents of every single node. We control this by means of removing

[1] Such model can be created manually, or by converting an existing Bayesian network to a bipartite graph. GeNIe in fact contains a function to this effect (see the *Diagnosis* menu).

weaker connections in the model. In the next section we speculate about more rigorous methods for modifying models once fielded.

6 Learning from just a Few Cases

It is natural to consider how automated learning can be applied in query-based diagnosis by using a selection of cases to modify an existing model. In this section we state this problem, and make a few observations about how one might go about it; the solution to it is outside the scope of this paper. The logging function of the system will generate supervised cases of session outcomes that can be used to improve the accuracy of the model. Much like a case-based reasoning system, the process envisioned consists of retrieving cases from a database when a new diagnosis starts. The process resembles conventional case-based reasoning, but enforces the strong consistency conditions of a Bayes network model.

Learning from cases is an additional way to exploit user work-flow, however unlike modifications made from elicitation of causal links, there is no direct way to modify the network by inspection of a set of cases. The problem is to determine if the set is consistent, and if so, to learn a model consistent with the set. To do this we offer a precise definition of probabilistic cases and case consistency:

Consider a database of cases indexed by j, corresponding to a selected set of diagnostic sessions that have been resolved. A case is the relevant part of the diagnostic state at the end of a diagnosis that has been validated, perhaps by replace-and-test, or by an expert's opinion. An individual case contains both evidence consisting of the complete set of observations from the session and the posterior of the top (or top few) faults.

Definition 1. Case. Given evidence for case j, $\mathbf{e}^{(j)} = \{e^{(j)}\}_{i=1\ldots I_j}$ and the posterior of faults $f^{(1)} \ldots f^{(k)}$ obtained by inference from the current model M_j, a case j is list of ordered fault marginals for that evidence: $\mathsf{P}\left(f^{(1)} \,|\, \mathbf{e}^{(j)}, M_j\right) \geq \ldots \geq \mathsf{P}\left(f^{(k)} \,|\, \mathbf{e}^{(j)}, M_j\right)$.

Definition 2. Case Consistency. A model M^* is consistent with a case j, to level k, if the list of ordered fault marginals given the evidence $\mathbf{e}^{(j)}$ agrees with the case: $\mathsf{P}\left(f^{(1)} \,|\, \mathbf{e}^{(j)}, M^*\right) \geq \ldots \geq \mathsf{P}\left(f^{(k)} \,|\, \mathbf{e}^{(j)}, M^*\right)$.

Clearly a case j is consistent with the model M_j that generated it.

6.1 Stochastic versus Epistemic Uncertainty

Given a large number of such cases, one may be tempted to apply conventional supervised learning techniques to refine the existing model. Putting aside the problem of a limited number of cases, and that the case consists most likely almost entirely of missing values, we argue that this is inappropriate from the user's point of view. The user considers a case as specifying the exact effect that they expect to see when the case is run on the correct model. The probabilities in the case are not variations due to a random sample, rather they indicate *epistemic uncertainty*—the knowledge, or lack of it in the diagnostic outcome given the observation set.

When a database of observations contains stochastic uncertainty, a statistical learning approach is appropriate, since, roughly, the true model is an average of many imprecise instances. In contrast, a case containing epistemic uncertainty is better considered as a fragment of the correct network model. The case is "supervised" in the sense of revealing a part of the joint distribution of the overall network.

The learning problem can now be stated as one of modifying an existing model to meet the set of constraints expressed by cases selected as relevant to the current diagnosis. We call this learning problem *case consistency*. Methods to solve this have been studied as applications of Jeffreys' Rule for belief revision (Chan and Darwiche, 2005). Such belief revision changes the joint distribution of the model. Jeffrey's rule is often presented as an alternate means of applying evidence, by comparison with applying likelihoods as evidence (sometimes called "virtual" or "soft" evidence, but the usage in the literature isn't consistent.) In contrast it appears preferable to distinguish the application of ev-

idence from case consistency, if only to make a clear semantic distinction between inference conditioned on observations, and an operation more like "splicing" a fragment of new knowledge into the existing model.

7 Conclusions

We described an approach to solving diagnostic problems that is based on the concept of *query-based diagnostics* and amounts to recovering a normative system based on Bayesian networks from a conventional diagnostic work-flow. Without forcing the user to use a limited vocabulary and variables, and by capturing the user's causal understanding, the system allows for continuous refinement of the model, while offering suggestions to avoid possible repetition of terms. While this carries with it a danger of uncontrolled model growth, we believe that with incorporation of workflow-generated cases for belief revision, and possibly off-line intervention of a knowledge engineer, the accuracy of the system can be maintained. Despite that the probabilistic model of a system under diagnosis is necessarily simplified, based on three-layer Bayesian networks with canonical interactions among the network variables, we are still able to reduce greatly the most important bottleneck—the knowledge engineering effort that goes into model building. We have two implementations of this idea underway, with preliminary results that are promising.

A system based on the principles outlined in this paper has to be thoroughly evaluated in a practical setting. One of our next steps is to employ our prototype in a diagnostic setting and carefully monitoring its use, including user experiences, model creation and growth, and the development of the system's diagnostic accuracy.

Acknowledgments

This work has been supported by Intel Research. Implementation of Marilyn is based on SMILE, a Bayesian inference engine developed at the Decision Systems Laboratory and available at `http://genie.sis.pitt.edu/`. We would like to thank Erik Pols for a considerable effort that went into developing initial ideas into a working MARILYN prototype and Parot Ratnapinda and his classmates at the University of Pittsburgh for refining Erik's initial implementation. We thank Cort Keller, Christine Matlock, Daniel Peters and Bryan Pollard for their work on design and implementation of Intel's application.

References

Avrim Blum. 1998. On-line algorithms in machine learning. In *Online Algorithms*, pages 306–325. Springer.

Hei Chan and Adnan Darwiche. 2005. On the revision of probabilistic beliefs using uncertain evidence. *Artif. Intell.*, 163(1):67–90.

Gregory F. Cooper. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artif. Intell.*, 42(2-3):393–405.

F. Javier Díez and Marek J. Druzdzel. 2008. Canonical probabilistic models for knowledge engineering. Unpublished manuscript, available at http://www.ia.uned.es/∼fjdiez/papers/canonical.html.

Eric Horvitz, Paul Koch, Carl M. Kadie, and Andy Jacobs. 2002. Coordinate probabilistic forecasting of presence and availability. In *18th Conference on Uncertainty in Artificial Intelligence*, pages 224–233. Morgan Kaufmann Publishers, July.

Kathryn B. Laskey. 2008. Mebn: A language for first-order bayesian knowledge bases. *Artif. Intell.*, 172(2-3):140–178.

Paul P. Maaskant and Marek J. Druzdzel. 2008. An Independence of Causal Interactions model for opposing influences. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM–08)*, Hirtshals, Denmark.

Guus Schreiber, Hans Schreiber, Anjo Akkermans, Robert de Anjewierden, Nigel Shadbolt Hoog, Walter Van de Velde, and Bob Wielinga. 2000. *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press.

S. Visweswaran and G.F. Cooper. 2004. Instance-specific bayesian model averaging for classification. In *Proceedings of the Neural Information Processing Systems Conference (NIPS-2004)*.

M. Wellman, J. Breese, and R. Goldman. 1992. From knowledge bases to decision models. *Knowledge Engineering Review*, 7(1):35–53.

High-dimensional probability density estimation with randomized ensembles of tree structured Bayesian networks

Sourour Ammar and Philippe Leray
Knowledge and Decision Team
Laboratoire d'Informatique de Nantes Atlantique (LINA) UMR 6241
Ecole Polytechnique de l'Université de Nantes, France
sourour.ammar@etu.univ-nantes.fr, philippe.leray@univ-nantes.fr

Boris Defourny and Louis Wehenkel
Department of Electrical Engineering and Computer Science & GIGA-Research,
University of Liège, Belgium
boris.defourny@ulg.ac.be, L.Wehenkel@ulg.ac.be

Abstract

In this work we explore the Perturb and Combine idea, celebrated in supervised learning, in the context of probability density estimation in high-dimensional spaces with graphical probabilistic models. We propose a new family of unsupervised learning methods of mixtures of large ensembles of randomly generated tree or poly-tree structures. The specific feature of these methods is their scalability to very large numbers of variables and training instances. We explore various simple variants of these methods empirically on a set of discrete test problems of growing complexity.

1 Introduction

Learning of Bayesian networks aims at modeling the joint density of a set of random variables from a random sample of joint observations of these variables (Cowell et al., 1999). Such a graphical model may be used for elucidating the conditional independences holding in the data-generating distribution, for automatic reasoning under uncertainties, and for Monte-Carlo simulations. Unfortunately, currently available algorithms for Bayesian network structure learning are either restrictive in the kind of distributions they search for, or of too high computational complexity to be applicable in very high dimensional spaces (Auvray and Wehenkel, 2008).

In the context of supervised learning, a generic framework which has led to many fruitful innovations is called "Perturb and Combine". Its main idea is to on the one hand perturb in different ways the optimization algorithm used to derive a predictor from a data-set and on the other hand to combine in some appropriate fashion a set of predictors obtained by multiple iterations of the perturbed algorithm over the data-set. In this framework, ensembles of weakly fitted randomized models have been studied intensively and used successfully during the last two decades. Among the advantages of these methods, let us quote the improved scalability of their learning algorithms and the improved predictive accuracy of their models. For example, ensembles of extremely randomized trees have been applied successfully in complex high-dimensional tasks, as image and sequence classification (Geurts et al., 2006).

In this work we explore the Perturb and Combine idea for probability density estimation. We study a family of learning methods to infer mixtures of large ensembles of randomly generated tree structured Bayesian networks. The specific feature of these methods is their scalability to very large numbers of variables and training instances. We explore various simple variants of these methods empirically on a set of discrete test problems of growing complexity.

The rest of this paper is organized as follows.

Section 2 discusses the classical Bayesian framework for learning mixtures of models. Section 3 describes the proposed approach and algorithms presented in this paper and Section 4 reports simulation results on a class of simple discrete test problems. Section 5 discusses our work in relation with the literature and Section 6 briefly concludes and highlights some directions for further research.

2 Bayesian modeling framework

Let $X = \{X_1, \ldots, X_n\}$ be a finite set of discrete random variables, and $D = (x^1, \cdots, x^d)$ be a data-set (sample) of joint observations $x^i = (x^i_1, \cdots, x^i_n)$ independently drawn from some data-generating density $\mathbb{P}_G(X)$.

In the full Bayesian approach, one assumes that $\mathbb{P}_G(X)$ belongs to some space of densities $\mathcal{D}$ described by a model-*structure* $M \in \mathcal{M}$ and model-*parameters* $\theta_M \in \Theta_M$, and one infers from the data-set a mixture of models described by the following equation:

$$\mathbb{P}_{\mathcal{D}}(X|D) = \sum_{M \in \mathcal{M}} \mathbb{P}(M|D)\ \mathbb{P}(X|M, D), \quad (1)$$

where $\mathbb{P}(M|D)$ is the posterior probability over the model-space $\mathcal{M}$ conditionally on the data D, and where $\mathbb{P}(X|M, D)$ is the integral:

$$\int_{\Theta_M} \mathbb{P}(X|\theta_M, M)\ d\mathbb{P}(\theta_M|M, D). \quad (2)$$

So $\mathbb{P}_{\mathcal{D}}(X|D)$ is computed by:

$$\sum_{M \in \mathcal{M}} \mathbb{P}(M|D) \int_{\Theta_M} \mathbb{P}(X|\theta_M, M)\ d\mathbb{P}(\theta_M|M, D), \quad (3)$$

where $d\mathbb{P}(\theta_M|M, D)$ is the posterior density of the model-parameter and $\mathbb{P}(X|\theta_M, M)$ is the likelihood of observation X for the structure M with parameter θ_M.

When the space of model-structures $\mathcal{M}$ is the space of Bayesian networks over X, approximations have to be done in order to make tractable the computation of Equation (3). (Chickering and Heckerman, 1997) show that Equation (2) can be simplified by the likelihood estimated with the parameters of maximum posterior probability $\tilde{\theta}_M = \arg\max_{\theta_M} \mathbb{P}(\theta_M|M, D)$, under the assumption of a Dirichlet distribution (parametrized by its coefficients α_i) for the prior distribution of the parameters $\mathbb{P}(\theta_M)$.

Another approximation to consider is simplifying the summation over all the possible model-structures M. As the size of the set of possible Bayesian network structures is super-exponential in the number of variables (Robinson, 1977), the summation of Equation (1) must be performed over a strongly constrained subspace $\hat{\mathcal{M}}$ obtained for instance by sampling methods (Madigan and Raftery, 1994; Madigan and York, 1995; Friedman and Koller, 2000), yielding the approximation

$$\mathbb{P}_{\hat{\mathcal{M}}}(X|D) = \sum_{M \in \hat{\mathcal{M}}} \mathbb{P}(M|D)\mathbb{P}(X|\tilde{\theta}_M, M). \quad (4)$$

Let us note here that this equation is simplified once more when using classical structure learning methods, by keeping only the model $M = \tilde{M}$ maximising $\mathbb{P}(M|D)$ over $\mathcal{M}$:

$$\mathbb{P}_{\tilde{M}}(X|D) = \mathbb{P}(X|\tilde{\theta}_{\tilde{M}}, \tilde{M}). \quad (5)$$

Let us emphasize that this further simplication has also the advantage of producing a *single* graphical model from which one can read of independencies *directly*. This may however be at the price of a possibly significant reduction of accuracy of the density estimation.

3 Randomized poly-tree mixtures

In this work, we propose to choose as set $\hat{\mathcal{M}}$ in Equation (4) a randomly generated subset of pre-specified cardinality of poly-tree models.

3.1 Poly-tree models

A poly-tree model for the density over X is defined by a directed acyclic graph structure P whose skeleton is acyclic and connected, and whose set of vertices is in bijection with X, together with a set of conditional densities $\mathbb{P}_P(X_i|pa_P(X_i))$, where $pa_P(X_i)$ denotes the set of variables in bijection with the parents of X_i in P. The structure P represents graphically the density factorization

$$\mathbb{P}_P(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbb{P}_P(X_i|pa_P(X_i)). \quad (6)$$

The model parameters are thus here specified by the set of distributions:

$$\theta_P = (\mathbb{P}_P(X_i|pa_P(X_i)))_{i=1}^n . \quad (7)$$

The structure P can be exploited for probabilistic inference over $\mathbb{P}_P(X_1, \ldots, X_n)$ with a computational complexity linear in the number of variables n (Pearl, 1986).

One can define nested subclasses $\mathcal{P}^p$ of poly-tree structures by imposing constraints on the maximum number p of parents of any node. In these subclasses, not only inference but also parameter learning is of linear complexity in the number of variables. The smallest such subclass is called the tree subspace, in which nodes have exactly one parent ($p = 1$).

When necessary, we will denote by $\mathcal{P}^*$ (respectively $\mathcal{P}^1$) the space of all possible poly-tree (respectively tree) structures defined over X.

3.2 Mixtures of poly-trees

A mixture distribution $\mathbb{P}_{\hat{\mathcal{P}}}(X_1, \ldots, X_n)$ over a set $\hat{\mathcal{P}} = \{P_1, \ldots, P_m\}$ of m poly-trees is defined as a convex combination of elementary poly-tree densities, i.e.

$$\mathbb{P}_{\hat{\mathcal{P}}}(X_1, \ldots, X_n) = \sum_{i=1}^{m} \mu_i \mathbb{P}_{P_i}(X_1, \ldots, X_n), \quad (8)$$

where $\mu_i \in [0, 1]$ and $\sum_{i=1}^m \mu_i = 1$, and where we leave for the sake of simplicity implicit the values of the parameter sets $\tilde{\theta}_i$ of the individual poly-tree densities.

While single poly-tree models impose strong restrictions on the kind of densities they can represent, mixtures of poly-trees are universal approximators, as well as mixtures of trees or chains, or even mixtures of empty graphs (i.e. Naive Bayes with hidden class), as shown in (Meila-Predoviciu, 1999) section 3.1.

3.3 Random poly-tree mixture learning

Our generic procedure for learning a random poly-tree mixture distribution from a data-set D is described by Algorithm 1; it receives as inputs X, D, m, and three procedures *DrawPolytree*, *LearnPars*, *CompWeights*.

Algorithm 1 (Learning a poly-tree mixture)

1. Repeat for $i = 1, \cdots, m$:
   (a) $P_i = \mathit{DrawPolytree}$,
   (b) For $j = 1, \cdots, n$:
   $\tilde{\theta}_{P_i} = \mathit{LearnPars}(P_i, D)$
2. $(\mu)_{i=1}^m = \mathit{CompWeights}((P_i, \tilde{\theta}_{P_i})_{i=1}^m, D)$
3. Return $\left(\mu_i, P_i, \tilde{\theta}_{P_i}\right)_{i=1}^m$.

3.4 Specific variants

In our first investigations reported below, we have decided to compare various simple versions of the above generic algorithm.

In particular, we consider both mixtures of randomly generated subsets of unconstrained poly-trees (by sampling from a uniform density over $\mathcal{P}^*$), and mixtures of tree structures (by sampling from a uniform density over $\mathcal{P}^1$). The random sampling procedures are described in Section 3.5.

As concerns the mixture coefficients, we will compare two variants, namely uniform weighting (coefficient $\mu_i = \frac{1}{m}, \forall i = 1, \ldots, m$) and Bayesian averaging (coefficient μ_i proportional to the posterior probability of the poly-tree structure P_i, derived from its BDeu score computed from the data-set (Cowell et al., 1999)).

Notice that with large data-sets, the Bayesian averaging approach tends to put most of the weight on the poly-tree which has the largest score; hence to better appraise the mixture effect, we will also provide results for the model which uses only the highest score structure among the m poly-trees of the ensemble, which amounts to a kind of random search for the MAP structure defined in Equation (5).

Finally, concerning parameter estimation, we use the BDeu score maximization for each poly-tree structure individually, which is tantamount to selecting the MAP estimates using Dirichlet priors. More specifically, in our experiments which are limited to binary random variables, we used non-informative priors, which then amounts to using $\alpha = 1/2$, i.e. $p(\theta, 1-\theta) \propto \theta^{-1/2}(1-\theta)^{-1/2}$ for the prior density of the parameters characterizing the conditional densities attached the poly-tree nodes.

3.5 Random generation of structures

Our mixture of random poly-trees and the experimental protocol described in Section 4 are based on random sampling of several classes of graphical structures (trees, poly-trees, and directed acyclic graphs).

For sampling trees and poly-trees, we chose to adapt the algorithm proposed by (Quiroz, 1989), which uses Prûfer coding of undirected tree structures. This algorithm allows to sample labelled undirected trees uniformly. We have adapted it in order to sample uniformly from the space of directed (rooted) trees and poly-trees. The resulting structure sampling algorithms are efficient, since their complexity remains linear in the number of variables. Notice however, that only in the case of tree structures these algorithms sample uniformly from the Markov equivalence classes induced by these structures. We do not know of any efficient adaptation of these algorithms to sample uniformly from structures of poly-trees with bounded number of in-degrees or to sample uniformly from Markov equivalence classes of poly-trees.

For sampling of directed acyclic graphs we used, on the other hand, the procedure given in (Ide et al., 2004), which allows to generate random structures which are of bounded in-degree and which are constrained to be connected. This scheme does neither yield a uniform sampling of these structures nor of their equivalence classes.

4 Preliminary empirical simulations

4.1 Protocol

For a first comparison of the different variants of our algorithm, we carried out repetitive experiments for different data-generating (or target) densities. All our experiments were carried out with models for a set of eight binary random variables. We chose to start our investigations in such a simple setting in order to be able to compute accuracies exactly (see Section 4.1.4), and so that we can easily analyze the graphical structures of the target densities and of the inferred set of poly-trees.

4.1.1 Choice of target density

To choose a target density $\mathbb{P}_G(X)$, we first decide whether it will factorize according to a poly-tree or to a directed acyclic graph structure. Then we use the appropriate random structure generation algorithm (see Section 3.5) to draw a structure and, we choose the parameters of the target density by selecting for each conditional density of the structure (they are all related to binary variables) two random numbers in the interval $[0, 1]$ and by normalizing.

4.1.2 Generation of data-sets

For each target density and data-set size, we generate 10 different data-sets by sampling values of the eight random variables using the Monte-Carlo method with the target structure and parameter values.

We carry out simulations with data-set sizes of 250 and 2000 elements respectively. Given the total number of 256 possible configurations of our eight random variables, we thus look at both small and large data-sets.

4.1.3 Learning of mixtures

For a given data-set and for a given variant of the mixture learning algorithm we generate ensemble models of growing sizes, respectively $m = 1$, $m = 10$, and then up to $m = 1000$ by increments of 10. This allows us to appraise the effect of the ensemble size on the quality of the resulting model.

4.1.4 Accuracy evaluation

The quality of any density inferred from a data-set is evaluated by the symmetric Kulback-Leibler divergence (Kullback and Leibler, 1951) between this density and the data-generating density $\mathbb{P}_G(X)$ used to generate the data-set. This is computed by

$$KL_s(\mathbb{P}_G, \mathbb{P}_M) = KL(\mathbb{P}_G||\mathbb{P}_M) + KL(\mathbb{P}_M||\mathbb{P}_G), \quad (9)$$

where $\mathbb{P}_M(X)$ denotes the density that is evaluated, and where

$$KL(\mathbb{P}||\mathbb{P}') = \sum_{X \in \mathcal{X}} \mathbb{P}(X) \ln \left(\frac{\mathbb{P}(X)}{\mathbb{P}'(X)} \right), \quad (10)$$

and $\mathcal{X}$ denotes the set of all possible configurations of the random variables in X.

We use this formula to evaluate our mixture models, and we also provide baseline values obtained with two different reference models, namely a *baseline* approach M_0 where a complete directed acyclic model is used with parameter values inferred by BDeu score maximization on the data-set, as well as a *golden standard* M_1 where the parameters of the target structure used to generate the data-set are re-estimated by BDeu score maximization from the data-set.

4.1.5 Software implementation

Our various algorithms of model generation were implemented in C++ with the Boost library available at `http://www.boost.org/` and various APIs provided by the ProBT© platform available at `http://bayesian-programming.org`.

4.2 Results

4.2.1 Sample of results

Figure 1 provides a representative set of learning curves for a target density corresponding to the directed acyclic graph (DAG) represented on he top of the figure. The middle and lower parts represent the learning curves obtained with respectively 250 and 2000 observations in the data-set. The horizontal axis corresponds to the number m of mixture terms, whereas the vertical axis corresponds to the KL_s measures with respect to the target density. All the curves represent average results obtained over ten different data-sets of the specified size.

The dashed horizontal lines in the lower parts of these graphics correspond to the golden standard M_1, whereas the plain horizontal line (not shown on the middle graphic) correspond to the M_0 baseline (its results are very bad on the small data-set and were therefore not shown).

The dashed, respectively black and red, curves in the upper part of both diagrams correspond to uniform mixtures of, respectively trees and poly-trees. We observe that their performances are quite disappointing, even though uniform poly-tree mixtures are slightly better than uniform tree mixtures.

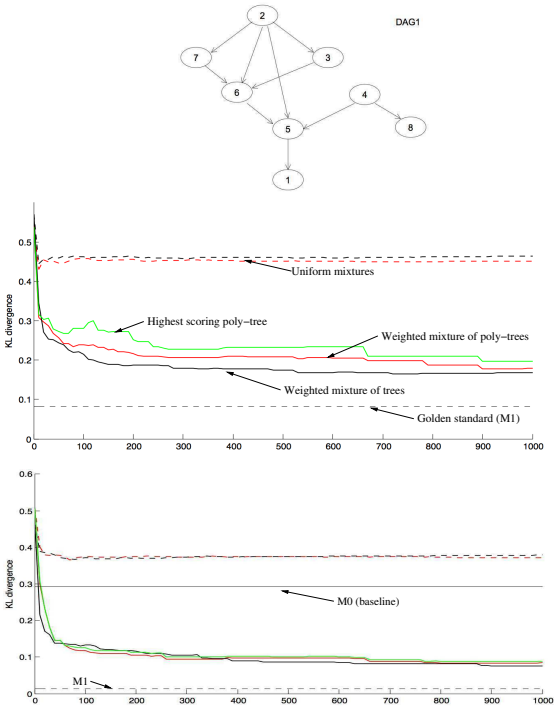


Figure 1: Example results. Top: target density DAG structure. Middle: learning curves with data-set size of 250 observation. Bottom: learning curves with 2000 observation. (see text for explanation of curves legends).

The two plain curves, respectively black and red, in the lower parts of the diagrams correspond to mixtures of respectively trees and poly-trees, when they are weighted proportionally to their posterior probability given the data-set. They provide much better performances, as compared to the baseline M_0 and are competitive with the golden standard M_1. We also observe that for the smaller sample size the tree mixtures outperform the poly-tree mixtures, whereas for the larger sample size they provide identical performances.

For the sake of comparison, we have also provided the behaviour of the "trivial mixture" (in green) which retains only the highest scoring structure of the generated ensemble. We observe that in small sample conditions, this latter model is outperformed by the plain Bayesian mixtures, while in the case of large sample size it is largely equivalent.

We complete these results with the two sets

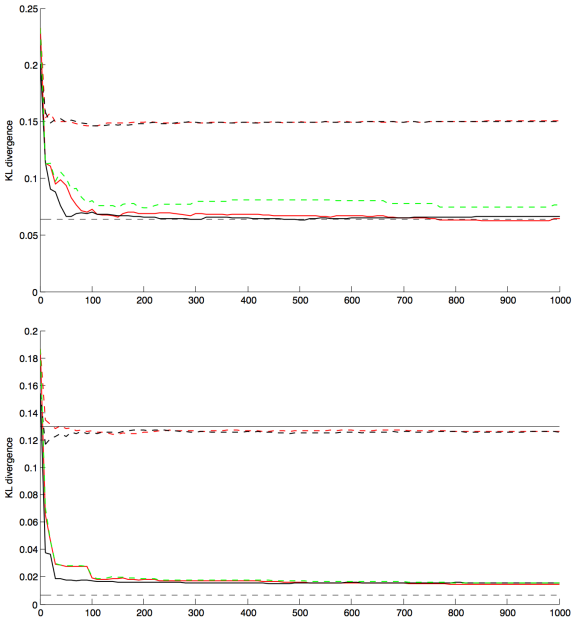


Figure 2: Example results for a target density with poly-tree structure.

of curves of Figure 2, obtained in similar conditions but when the target density factorizes according to a poly-tree structure. Overall, the previous conclusions still hold true. The main difference that we observe, is that in the case of the poly-tree target density the KL_s scores seem to converge more rapidly towards M_1 when the mixture size m increases.

4.2.2 Analysis of asymptotic behavior

Since in most trials, our learning curves stabilized around $m = 1000$, we consider interesting to provide a synthetic picture of the performances of the different methods under these "asymptotic" conditions. To this end, we show on Figure 3 overall asymptotic performances in the form of box plots of the KL_s values of the different methods.

On this figure, each box plot (box-and-whisker diagram) depicts the density of KL_s values of a particular algorithm variant (from left to right the golden standard M_1, the weighted poly-tree mixtures and the weighted tree mixtures), for a fixed sample size, but over the combination of 5 different target DAG structures, 3 different target poly-tree structures, and for each case 10 data-sets. For the sake

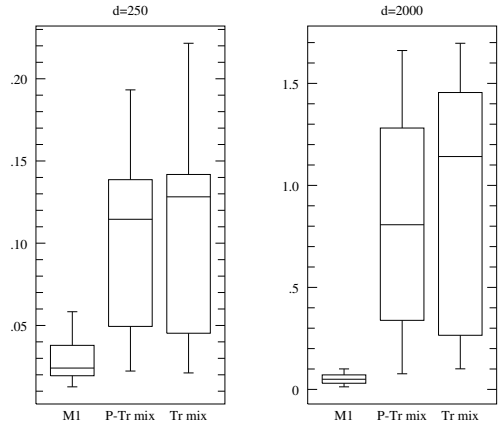


Figure 3: Synthesis of asymptotic behavior ($m = 1000$, relative KL_s wrt baseline M_0).

of interpretation, the KL_s values are normalized by dividing them by the value obtained in the same conditions (same target density, and same data-set) by the M_0 baseline method (and the latter values are not represented on the box-plots). In the left part we show the results obtained with a sample size of $d = 250$ and in the right part with a sample of size $d = 2000$.

We can synthesize these results as follows. For both large and small sample sizes the poly-tree mixtures outperform (but only very slightly) the tree mixtures; this effect is less notable in small sample conditions. In large sample conditions ($d = 2000$), the poly-tree mixtures have only a small advantage over the baseline method M_0 (their relative scores being on the average only slightly smaller than 1). However, in small sample conditions ($d = 250$), both poly-tree and tree mixtures are significantly better than the baseline, and, actually they yield KL_s scores which are already quite close to those of the the golden standard M_1.

5 Discussion

Our choice of using random mixtures of poly-trees was inspired by several considerations.

First of all, choosing the best structure in the space of poly-trees is not an adequate solution from an algorithmic point of view. Indeed, (Dasgupta, 1999) shows that finding the optimal poly-tree model is not tractable for very high dimensional spaces. On the other hand, the space of poly-trees is a priori a rather rich

space, and it is characterized by efficient inference algorithms. Hence, even a very large mixture of poly-tree densities can be queried efficiently for making inferences about the data-generating density. Furthermore, using mixtures of poly-trees allows in principle to represent any density.

In our experiments on the very simple problems with 8 binary variables, we observed however that in most cases using a mixture of poly-trees was not really better than keeping only the single best found poly-tree (except in very small sample size conditions).

Our second reason for looking at poly-tree mixtures was that we thought that these models would be more powerful than tree mixtures. Indeed, (Meila-Predoviciu, 1999) already proposed to use mixtures of tree models and has designed algorithms to find the optimal combination of tree structures and of the coefficients of the mixture during the learning phase. She jointly uses the MWST (Maximum Weight Spanning Tree) structure learning algorithm published in the late sixties (Chow and Liu, 1968) and the Expectation-Maximization algorithm for coefficients' estimation (Dempster et al., 1977). While this proposal is very elegant, we believe that it is not scalable to very large mixtures, both from the point of view of computational complexity and from the point of view of risk of over-fitting the data-set.

Our simulation results showed however that using random mixtures of poly-trees is only very marginally advantageous with respect to the use of random mixtures of trees. On the other hand, in small sample conditions the mixtures of trees or poly-trees turned out to be quite often of comparable accuracy than the golden standard $M1$, and in general largely superior to the complete structure baseline M_0.

Concerning the weighting scheme, our experiments also confirmed that uniform mixtures of randomized poly-tree or tree structured densities do not work properly in the context of density estimation. This is quite different from the observations made in the context of tree-based supervised learning, where uniform mixtures of totally randomized trees often provide very competitive results (Geurts et al., 2006). The main difference between these two contexts is that in supervised learning one can easily generate a sample of randomized trees which fit well the data-set, whereas in the context of density estimation random tree or poly-tree structures mostly strongly under-fit the data-set.

6 Summary and future works

We have proposed in this paper to transpose the "Perturb and Combine" idea celebrated in supervised learning to density estimation. We have presented a generic framework for doing this, based on random mixtures of poly-tree or tree structured Bayesian networks.

The first results obtained in the context of a simple test protocol are already interesting, while they also highlight a certain number of immediate future research directions.

Thus, a first line of research will be to apply our experimental protocol to a larger set of problems including high-dimensional ones and a larger range of sample sizes. We believe also that a more in depth analysis of the results with respect to the basic properties of the target distributions would be of interest. Of course, these investigations should also aim at systematically comparing all these algorithm variants both from a computational complexity and from an accuracy point of view with other mixture models proposed in the literature (Meila-Predoviciu, 1999; Lowd and Domingos, 2005), with state-of-the-art optimal structure learning algorithms (Auvray and Wehenkel, 2008), and with other approaches proposed for efficient learning (Friedman et al., 1999; Brown et al., 2004) and/or efficient inference (Jaeger, 2004) in high dimensional spaces.

Nevertheless, from these first results we are tempted to conclude that, in order to effectively transpose the Perturb and Combine idea to the context of density estimation, it will be necessary to design structure sampling algorithms which are able to efficiently focus on structures that can be fitted well enough to the available data-set. In this respect, one straightforward idea would be to transpose the Bagging idea of (Breiman, 1996) to the density estimation

context. In particular, we suggest that the use of bootstrap sampling in combination with the efficient algorithm of finding the optimal tree model, i.e. solving Equation (5) in the tree space $\mathcal{P}^1$ using the MWST algorithm, could be a very promising direction.

Another more generic direction of research, is to adapt importance sampling approaches (e.g. the cross-entropy method (Rubinstein and Kroese, 2004)) in order to generate randomized ensembles of simple structures (trees, polytrees, etc.) that fit well the given data-set.

In a later stage, we intend to extend these algorithms to the case of continuous random variables as well as when there are missing data.

Acknowledgments

This work presents research results of the Belgian Network BIOMAGNET (Bioinformatics and Modeling: from Genomes to Networks), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office.

References

V. Auvray and L. Wehenkel. 2008. Learning inclusion-optimal chordal graphs. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI-2008)*.

L. Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

L. E. Brown, I. Tsamardinos, and C. F. Aliferis. 2004. A novel algorithm for scalable and accurate bayesian network learning. *Medinfo*, 11(Pt 1):711–715.

D.M. Chickering and D. Heckerman. 1997. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212.

C.K. Chow and C.N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer.

S. Dasgupta. 1999. Learning polytrees. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 134–14, San Francisco, CA. Morgan Kaufmann.

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. Series B (Methodological)*, 39(1):1–38.

N. Friedman and D. Koller. 2000. Being Bayesian about network structure. In C, editor, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 201–210, SF, CA, June 30–July 3. Morgan Kaufmann Publishers.

N. Friedman, I. Nachman, and D. Pe'er. 1999. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 206–215. Morgan Kaufmann.

P. Geurts, D. Ernst, and L. Wehenkel. 2006. Extremely randomized trees. *Machine Learning*, 63(1):3–42.

J.S. Ide, F.G. Cozman, and F.T. Ramos. 2004. Generating random bayesian networks with constraints on induced width. In *ECAI*, pages 323–327.

M. Jaeger. 2004. Probabilistic decision graphs - combining verification and AI techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(Supplement-1):19–42.

S. Kullback and R. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

D. Lowd and P. Domingos. 2005. Naive bayes models for probability estimation. In *Proceedings of the Twenty-Second International Conference (ICML 2005)*, pages 529–536. ACM.

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.

D. Madigan and J. York. 1995. Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232.

M. Meila-Predoviciu. 1999. *Learning with Mixtures of Trees*. Ph.D. thesis, MIT.

J. Pearl. 1986. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288.

A. Quiroz. 1989. Fast random generation of binary, t-ary and other types of trees. *Journal of Classification*, 6(1):223–231, December. available at http://ideas.repec.org/a/spr/jclass/v6y1989i1p223-231.html.

R.W. Robinson. 1977. Counting unlabeled acyclic digraphs. In C. H. C. Little, editor, *Combinatorial Mathematics V*, volume 622 of *Lecture Notes in Mathematics*, pages 28–43, Berlin. Springer.

R.Y. Rubinstein and D.P. Kroese. 2004. *The Cross-Entropy Method. A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Information Science and Statistics. Springer.

Generalized Loopy 2U: A New Algorithm for Approximate Inference in Credal Networks

Alessandro Antonucci, Marco Zaffalon, Yi Sun, Cassio P. de Campos
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Lugano, Switzerland

Abstract

Credal nets generalize Bayesian nets by relaxing the requirement of precision of probabilities. Credal nets are considerably more expressive than Bayesian nets, but this makes belief updating NP-hard even on polytrees. We develop a new efficient algorithm for approximate belief updating in credal nets. The algorithm is based on an important representation result we prove for general credal nets: that any credal net can be equivalently reformulated as a credal net with binary variables; moreover, the transformation, which is considerably more complex than in the Bayesian case, can be implemented in polynomial time. The equivalent binary credal net is updated by L2U, a loopy approximate algorithm for binary credal nets. Thus, we generalize L2U to non-binary credal nets, obtaining an accurate and scalable algorithm for the general case, which is approximate only because of its loopy nature. The accuracy of the inferences is evaluated by empirical tests.

1 Introduction

Bayesian nets (Sect. 2.1) are probabilistic graphical models based on precise assessments for the conditional probability mass functions of the net variables given the values of their parents. As a relaxation of such precise assessments, *credal nets* (Sect. 2.2) only require the conditional probability mass functions to belong to convex sets of mass functions, i.e., *credal sets*. Credal nets (CNs) are considerably more expressive than Bayesian nets,[1] and the price is an increased complexity of inference: *belief updating* in credal nets is NP-hard even on polytrees (de Campos and Cozman, 2005). The only known exception to this situation is the algorithm *2U* (Fagiuoli and Zaffalon, 1998), as it computes exact posterior beliefs on *binary* (i.e., with binary variables) polytree-shaped CNs in linear time. A *loopy* version of 2U (L2U) has been proposed for multiply connected binary credal nets by (Ide and Cozman, 2004). Inferences based on L2U are approximate, but a good accuracy is typically observed after few iterations (Sect. 2.3).

In this paper we develop an efficient algorithm for approximate belief updating of general CNs (any topology, number of states per variable). The algorithm is based on an important representation result that we prove in Appendix A: that any CN can be equivalently reformulated as one with binary variables. The corresponding transformation, which is considerably more complex than in the Bayesian case, is based on two distinct transformations: a *decision-theoretic specification* (Antonucci and Zaffalon, 2008), which augments the CN with control variables enumerating the multiple mass functions owned by the nodes of the net (Sect. 3.2); a *binarization* procedure (Antonucci et al., 2006) that transforms each variable into a cluster of binary variables (Sect. 3.1).

We prove that the sequential application of these two transformations, originally developed for independent reasons, returns an equiva-

[1] Greater expressiveness is a consequence of the fact that Bayesian nets are a subset of credal nets. Expressiveness should not be confused with informativeness: for example, it is thanks to the greater expressiveness that credal nets can model much less informative states of knowledge (including lack of knowledge) than those Bayesian nets can model.

lent binary representation of the original CN (Sect. 4.1). Such equivalent binary CN can be finally updated by L2U. Overall, that leads to a *generalized loopy 2U* (GL2U) algorithm for the updating in general CNs, whose only source of approximation is the loopy part (Sect. 4.2). The algorithm, which takes polynomial time (Sect. 4.3), has been implemented in a software. Experimental tests in Sect. 5 show that its accuracy is comparable to that of state-of-the-art approximate methods for CNs. This, together with its scalability, should make of GL2U the algorithm of choice especially for large nets.

2 Bayesian and Credal Nets

In this section we review the basics of Bayesian nets (BNs) and their extension to convex sets of probabilities, i.e., credal nets. Both the models are based on a collection of random variables, structured as a vector $\mathbf{X} := (X_1, \ldots, X_n)$,[2] and a directed acyclic graph (DAG) $\mathcal{G}$, whose nodes are associated with the variables of $\mathbf{X}$. In our assumptions the variables in $\mathbf{X}$ take values in finite sets. For both models, 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. What makes BNs and CNs different is a different notion of independence and a different characterization of the conditional mass functions for each variable given the values of the parents, which will be detailed later.

Regarding notation, for each $X_i \in \mathbf{X}$, $\Omega_{X_i} = \{x_{i0}, x_{i1}, \ldots, x_{i(d_i-1)}\}$ denotes the possibility space of X_i, $P(X_i)$ is a mass function for X_i and $P(x_i)$ the probability that $X_i = x_i$, where x_i is a generic element of Ω_{X_i}. A similar notation with uppercase subscripts (e.g., X_E) denotes vectors (and sets) of variables in $\mathbf{X}$. Finally, the parents of X_i, according to $\mathcal{G}$, are denoted by Π_i, while for each $\pi_i \in \Omega_{\Pi_i}$, $P(X_i|\pi_i)$ is the mass function for X_i conditional on $\Pi_i = \pi_i$.

[2]The symbol ":=" is used for definitions.

2.1 Bayesian nets

For BNs, a conditional mass function $P(X_i|\pi_i)$ for each $X_i \in \mathbf{X}$ and $\pi_i \in \Omega_{\Pi_i}$ should be defined; and the standard notion of probabilistic independence is assumed in the Markov condition. A BN can therefore be regarded as a joint probability mass function over $\mathbf{X}$ that, according to the Markov condition, factorizes as follows:

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

for all the possible values of $\mathbf{x} \in \Omega_{\mathbf{X}}$, with the values of x_i and π_i consistent with $\mathbf{x}$. In the following, we represent a BN as a pair $\langle \mathcal{G}, P(\mathbf{X}) \rangle$. Posterior beliefs about a queried variable X_q, given evidence $X_E = x_E$, are defined as:

$$P(x_q|x_E) = \frac{\sum_{x_M} \prod_{i=1}^{n} P(x_i|\pi_i)}{\sum_{x_M, x_q} \prod_{i=1}^{n} P(x_i|\pi_i)}, \tag{2}$$

where $X_M := \mathbf{X} \setminus (\{X_q\} \cup X_E)$, the domains of the arguments of the sums are left implicit and the values of x_i and π_i are consistent with $\mathbf{x} = (x_q, x_M, x_E)$. Evaluating Eq. (2) is an NP-hard task, but for polytrees, Pearl's propagation allows for efficient updating (Pearl, 1988).

2.2 Credal sets and credal nets

CNs relax BNs by allowing for *imprecise probability* statements: in our assumptions, the conditional mass functions of a CN are required to belong to a *finitely generated* credal set, i.e., the convex hull of a finite number of mass functions for a certain variable. 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 corresponding to the *vertices* of the polytope. Updating based on a credal set is equivalent to that based only on its vertices (Walley, 1991). A credal set over X will be denoted as $K(X)$ and the set of its vertices as $\mathrm{ext}[K(X)]$. Given a non-empty $\Omega_X^* \subseteq \Omega_X$, an important credal set for our purposes is the *vacuous credal set* relative to Ω_X^*, i.e., the set of all the mass functions for X assigning probability one to Ω_X^*. We denote this set by $K_{\Omega_X^*}(X)$. In the following

we will use the well-known fact that the vertices of $K_{\Omega^*_X}(X)$ are the[3] $|\Omega^*_X|$ degenerate mass functions assigning probability one to the single elements of Ω^*_X. Marginalization generalizes to credal sets as follows: the marginalization $K(X)$ of a joint credal set $K(X,Y)$ to X is the convex hull of the mass functions $P(X)$ obtained from the marginalization of $P(X,Y)$ to X for each $P(X,Y) \in K(X,Y)$.

In order to specify a CN over the variables in $\mathbf{X}$ based on $\mathcal{G}$, a collection of conditional credal sets $K(X_i|\pi_i)$, one for each $\pi_i \in \Omega_{\Pi_i}$, should be provided separately for each $X_i \in \mathbf{X}$; while, regarding the Markov condition, we assume *strong independence* (Cozman, 2005). A CN associated with these local specifications is said to be with *separately specified* credal sets. Fig. 1 reports a CN, whose specification requires the (separate) assessment of an unconditional credal set for X_1, and two and eight conditional credal sets for X_2 and X_3. The specification becomes global considering the *strong extension* $K(\mathbf{X})$ of the CN, i.e., the convex hull of the following collection of joint mass functions:

$$\left\{ \prod_{i=1}^{n} P(X_i|\Pi_i) : P(X_i|\pi_i) \in K(X_i|\pi_i) \quad \begin{array}{l} \forall \pi_i \in \Omega_{\Pi_i}, \\ \forall i = 1,\ldots,n \end{array} \right\}. \tag{3}$$

We represent a CN as a pair $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$, where $\mathbf{P}(\mathbf{X}) := \{P_k(\mathbf{X})\}_{k=1}^{n_v} := \text{ext}[K(\mathbf{X})]$. Clearly, for each $k = 1,\ldots,n_v$, $\langle \mathcal{G}, P_k(\mathbf{X}) \rangle$ is a BN. For this reason a CN can be regarded as a finite set of BNs. For CNs *updating* is intended as the computation of tight bounds of the posterior probabilities of a queried variable given some evidence, i.e., Eq. (2) generalizes as:

$$\underline{P}(x_q|x_E) = \min_{k=1,\ldots,n_v} \frac{\sum_{x_M} \prod_{i=1}^{n} P_k(x_i|\pi_i)}{\sum_{x_M,x_q} \prod_{i=1}^{n} P_k(x_i|\pi_i)}, \tag{4}$$

and similarly for upper probabilities $\overline{P}(x_q|x_E)$. Exact updating in CNs displays high complexity. Updating in polytree-shaped CNs is NP-complete, and NP^{PP}-complete in general CNs (de Campos and Cozman, 2005). The only known exact linear-time algorithm for updating a specific class of CNs is the 2U algorithm, which we review in the following section.

[3]The cardinality of a set Ω is denoted as $|\Omega|$.

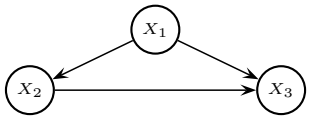


Figure 1: A separately specified CN over (X_1, X_2, X_3), with $|\Omega_{X_1}|=2$, $|\Omega_{X_2}|=|\Omega_{X_3}|=4$.

2.3 2U and its loopy extension

The adaptation of Pearl's updating algorithm to polytree-shaped binary CNs led to an exact algorithm called *2-Updating* (2U) (Fagiuoli and Zaffalon, 1998). Remarkably, 2U updates binary polytrees in time linear in the input size.

Loopy propagation is a popular technique that applies Pearl's propagation to multiply connected BNs: propagation is iterated until probabilities converge or for a fixed number of iterations. A loopy variant of 2U (L2U) can update multiply connected binary CNs in an approximate way (Ide and Cozman, 2004). Initialization of variables and messages follows the same steps used in the 2U algorithm. Then nodes are repeatedly updated, until convergence of probabilities is observed.[4] L2U is basically an iteration of 2U and its complexity is therefore linear in the input size and in the number of iterations. Overall, the L2U algorithm is an excellent algorithm, it is fast and returns good results with low errors after a few iterations.

3 Transformations of Credal Nets

In this section we review two different transformations of CNs that have been recently proposed for independent reasons. Their sequential application is the basis to obtain an equivalent representation of CNs based on binary variables.

3.1 Binarization algorithm

By definition, L2U (see Sect. 2.3) cannot be applied to a non-binary CN in the example of Fig. 1. To overcome this limitation, a *binarization* that transforms a CN into a binary CN has been proposed in (Antonucci et al., 2006).

First, each variable is equivalently represented by a cluster of binary variables. Assume

[4]Despite the lack of a formal proof, convergence of L2U has been always observed in the numerical tests.

d_i, which is the number of states for X_i, to be an integer power of two, and let $\tilde{d}_i := \log_2 |\Omega_{X_i}|$.[5] An obvious one-to-one correspondence between the states of X_i and the joint states of a vector of $\tilde{d}_i$ binary variables $\tilde{X}_i := (\tilde{X}_i^0, \tilde{X}_i^1, \ldots, \tilde{X}_i^{\tilde{d}_i-1})$ is established if the joint state $(\tilde{x}_i^0, \ldots, \tilde{x}_i^{\tilde{d}_i-1}) \in \{0,1\}^{\tilde{d}_i}$ is associated with $x_{il} \in \Omega_{X_i}$, where l is the integer whose $\tilde{d}_i$-bit representation is $\tilde{x}_i^{\tilde{d}_i-1} \cdots \tilde{x}_i^1 \tilde{x}_i^0$. Elements of $\tilde{X}_i$ are said *bits* of X_i and their position in the vector their *order*.

Overall, $\tilde{\mathbf{X}}$ denotes the vector of bits obtained *binarizing* all the elements of $\mathbf{X}$. We write $P(\mathbf{X}) = \tilde{P}(\tilde{\mathbf{X}})$, if $P(\mathbf{x}) = \tilde{P}(\tilde{\mathbf{x}})$ for each $\mathbf{x} \in \Omega_{\mathbf{X}}$, where $\tilde{\mathbf{x}} \in \Omega_{\tilde{\mathbf{X}}}$ is the state corresponding to $\mathbf{x}$.

A DAG $\tilde{\mathcal{G}}$ associated to the variables $\tilde{\mathbf{X}}$ can be obtained from $\mathcal{G}$ as follows: (i) two nodes of $\tilde{\mathcal{G}}$ corresponding to bits of different variables in $\mathbf{X}$ are connected by an arc if and only if there is an arc with the same direction between the related variables in $\mathbf{X}$; (ii) an arc connects two nodes of $\tilde{\mathcal{G}}$ corresponding to bits of the same variable of $\mathbf{X}$ if and only if the order of the bit associated to the node from which the arc departs is lower than the order of the bit associated to the remaining node. An example of this transformation is depicted in Fig. 2.

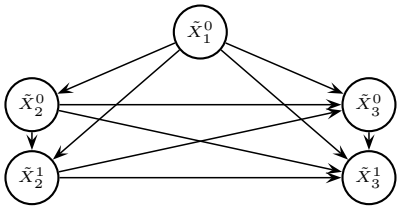


Figure 2: The binarization of the CN in Fig. 1.

Finally, regarding the quantification of the conditional credal sets, we have:

$$\underline{\tilde{P}}(\tilde{x}_i^j|\tilde{\pi}_i^j) := \min_{k=1,\ldots,n_v} \tilde{P}_k(\tilde{x}_i^j|\tilde{\pi}_i^j), \qquad (5)$$

where the index k is defined like in Eq. (4). Denoting by $\tilde{\Pi}_i$ the parents of $\tilde{X}_i^j$ corresponding to the binarization of Π_i, i.e., those that are not in the same cluster of $\tilde{X}_i^j$, the probabilities to be minimized on the right-hand side are:

$$\tilde{P}_k(\tilde{x}_i^j|\tilde{x}_i^{j-1}, \ldots, \tilde{x}_i^0, \tilde{\pi}_i) \propto \sum\nolimits_l^* P_k(x_{il}|\pi_i), \qquad (6)$$

where the sum $\sum^*$ is restricted to the states $x_{il} \in \Omega_{X_i}$ such that $l \bmod 2^{j+1}$ is the integer whose $(j+1)$-bit representation is $\tilde{x}_i^j, \ldots, \tilde{x}_i^1, \tilde{x}_i^0$, π_i is the joint state of the parents of X_i corresponding to the joint state $\tilde{\pi}_i$ for the bits of the parents of X_i, symbol $\propto$ denotes proportionality, and the relations are considered for each $i = 1, \ldots, n$, $j = 0, \ldots, \tilde{d}_i - 1$, and $\pi_i \in \Omega_{\Pi_i}$.

If both the states of $\tilde{X}_i^j$ produce zero in Eq. (6), the corresponding conditional mass functions can be arbitrarily specified (we set a degenerate mass function). Note that minimization in Eq. (5) can be obtained by simply considering the vertices of $K(X_i|\pi_i)$ in Eq. (6).

The overall procedure returns a well-defined CN, which is called the *binarization* of the original CN. Given an updating problem on a CN as in Eq. (4), we can consider the corresponding problem on its binarization. E.g., the computation of $\underline{P}(x_{33}|x_{10})$ for the CN in Fig. 1 corresponds to $\underline{\tilde{P}}(\tilde{X}_3^0 = 1, \tilde{X}_3^1 = 1|\tilde{X}_1^0 = 0)$. According to (Antonucci et al., 2006, Th. 2) this is an *outer approximation* (i.e., the posterior interval includes that of the original updating problem), which can be approximately estimated by L2U.

This approach entails a twofold approximation: (i) the approximation introduced by the binarization and (ii) that due to the loopy propagation. Approximation (i) can be regarded as originated by replacing each credal set of the original net with an enclosing polytope that can have a smaller number of vertices. By construction, the latter number cannot be controlled and could be too low to lead to a satisfactory approximation of the original credal set, which in turns leads approximation (i) to be quite crude. In the next section, we recall an independently developed transformation that will be used to remove approximation (i).

3.2 Decision-theoretic specification

In (Antonucci and Zaffalon, 2008), a general graphical language for CNs based on the so-called *decision-theoretic specification* (DTS) has

[5] This is not a limitation as a number of *dummy* states up the the nearest power of two can be always added. From now on we assume for all the variables a number of possible values equal to an integer power of two.

been proposed. A DTS of a CN is obtained augmenting the original CN by a number of *control nodes*, used to enumerate the vertices of the conditional credal sets. That turns the original nodes into precise-probability ones, while the control nodes can be formulated as chance nodes with vacuous credal sets.

Let us briefly describe this transformation in the case of a CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$. First, we obtain from $\mathcal{G}$ a second DAG $\mathcal{G}'$ defined over a wider domain $\mathbf{X}' := (X_1, \ldots, X_{2n})$. This is done by iterating, for each $i = 1, \ldots, n$, the following operations: (i) add a node X_{i+n}; (ii) draw an arc from each parent of X_i to X_{i+n}; (iii) delete the arcs connecting the parents of X_i with X_i; (iv) draw an arc from X_{i+n} to X_i. An example of this transformation is shown in Fig. 3.

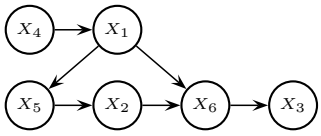


Figure 3: The output of the transformation described in Sect. 3.2 for the CN in Fig. 1.

Note that, for each $i = 1, \ldots, n$, $\Pi'_{i+n} = \Pi_i$, i.e., the parents of X_{i+n} in $\mathcal{G}'$ are the parents of X_i in $\mathcal{G}$, and also $\Pi'_i = X_{i+n}$, i.e., X_{i+n} is the only parent of X_i in $\mathcal{G}'$ and is therefore called the *control variable* of X_i.

We assume a one-to-one correspondence between the possible states of a control variable X_{i+n} and the collection of all the (distinct) extreme mass functions of *all* the conditional credal sets specified over X_i, i.e., $\Omega_{X_{i+n}} := \bigcup_{\pi_i \in \Omega_{\Pi_i}} \text{ext}[K(X_i|\pi_i)]$, for each $i = 1, \ldots, n$. As an example, assuming the number of vertices for the credal sets of the CN in Fig. 1 equal to the number of possible states of the relative variables, we have that X_4 in Fig. 3 is a binary variable, whose states correspond to the two vertices of $K(X_1)$; X_5 has eight possible states corresponding to the four vertices of $K(X_2|x_1)$ and the four of $K(X_2|\neg x_1)$; X_6 has 32 possible states corresponding to the vertices, four per each set, of the conditional credal sets over X_3.

Finally, in order to obtain a well-defined CN over $\mathbf{X}'$ associated to $\mathcal{G}'$, we quantify the conditional credal sets as follows. For each $i = 1, \ldots, n$, we set $K'(X_i|x_{i+n}) := P(X_i)_{x_{i+n}}$, where $P(X_i)_{x_{i+n}}$ is the element of $\text{ext}[K(X_i|\pi_i)]$ corresponding to x_{i+n}. For the control nodes $\{X_{i+n}\}_{i=1}^n$, we set $K'(X_{i+n}|\pi'_{i+n}) := K_{\Omega^{\pi_i}_{X_{i+n}}}(X_i)$, where $\Omega^{\pi_i}_{X_{i+n}} \subseteq \Omega_{X_{i+n}}$ is the set of vertices of $K(X_i|\pi_i)$.

The CN returned by this transformation will be denoted as $\langle \mathcal{G}', \mathbf{P}'(\mathbf{X}') \rangle$, and its strong extension as $K'(\mathbf{X}')$. Remarkably, $\langle \mathcal{G}', \mathbf{P}'(\mathbf{X}') \rangle$ provides an equivalent representation of $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$ being that $K'(\mathbf{X}) = K(\mathbf{X})$ as stated by Th. 2 in (Antonucci and Zaffalon, 2008), where $K'(\mathbf{X})$ is the marginalization of $K'(\mathbf{X}')$ to $\mathbf{X}$.

4 Exact Binarization & GL2U

Now we present the original contributions of this paper, consisting of a general representation result (Sect. 4.1), the definition of GL2U (Sect. 4.2), the study of its computational complexity, and its empirical evaluation (Sect. 5).

4.1 Exact binarization

Consider the sequential application of the transformations detailed in Sect. 3.2 and Sect. 3.1. Thus, given a CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$, obtain $\langle \mathcal{G}', \mathbf{P}'(\mathbf{X}') \rangle$ by a DTS, and hence $\langle \tilde{\mathcal{G}}', \tilde{\mathbf{P}}'(\tilde{\mathbf{X}}') \rangle$ through binarization. The latter CN is said the *exact binarization* of the first, a terminology justified by the following result.

Theorem 1. *Consider a CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$ and its exact binarization $\langle \tilde{\mathcal{G}}', \tilde{\mathbf{P}}'(\tilde{\mathbf{X}}') \rangle$. Let $K(\mathbf{X})$ and $\tilde{K}'(\tilde{\mathbf{X}}')$ be their corresponding strong extensions. Then:*

$$K(\mathbf{X}) = \tilde{K}'(\tilde{\mathbf{X}}), \tag{7}$$

with $\tilde{K}'(\tilde{\mathbf{X}})$ marginalization of $\tilde{K}'(\tilde{\mathbf{X}}')$ to $\tilde{\mathbf{X}}$.

According to Eq. (7), $\langle \tilde{\mathcal{G}}', \tilde{\mathbf{P}}'(\tilde{\mathbf{X}}') \rangle$ is an equivalent binary representation of $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$. It should be pointed out that, even if we focus on the case of CNs with separately specified credal sets, Th. 1 holds also for so-called *non-separately specified* CNs, for which a DTS can be provided as well. Similarly, the algorithm presented in the next section can be applied to any CN, separately or non-separately specified.

4.2 GL2U

Th. 1 is a basis for the solution of general inference problems, as stated by the following straightforward corollary.

Corollary 1. *Any inference problem on a CN can be equivalently computed in its exact binarization.*

According to Cor. 1, we can consider a so-called *generalized* L2U algorithm (GL2U), where given an updating problem on a CN, we solve by L2U the corresponding updating problem on the exact binarization of the original CN. The overall procedure is still approximate, but differently from the case without DTS considered in (Antonucci et al., 2006), the only source of approximation is the loopy component.

4.3 Complexity issues

According to the discussion in the previous section, the computational time required by GL2U to update a CN $\langle \mathcal{G}, \mathbf{P}(\mathbf{X}) \rangle$ is basically that required by L2U to update $\langle \tilde{\mathcal{G}}', \tilde{\mathbf{P}}'(\tilde{\mathbf{X}}') \rangle$. This is $O(t \cdot 2^{2a})$, where t is the number of iterations and a the maximum indegree of $\mathcal{G}'$. It can be checked that $\tilde{X}_i^{\tilde{d}_i - 1}$ has the maximum indegree among the $\tilde{d}_i$ binary nodes in the cluster $\tilde{X}_i$; similarly, $\tilde{X}_{i+n}^{\tilde{d}_{i+n}-1}$ has the maximum indegree among the $\tilde{d}_{i+n}$ nodes of $\tilde{X}_{i+n}$. Note also that the number of nodes in $\tilde{\Pi}_i$ is $\sum_{j/X_j \in \Pi_i} \tilde{d}_j$. Therefore, the indegrees of $\tilde{X}_i^{\tilde{d}_i - 1}$ and $\tilde{X}_{i+n}^{\tilde{d}_{i+n}-1}$ are respectively $\tilde{d}_i + \tilde{d}_{i+n} - 1$ and $\tilde{d}_{i+n} + \sum_{j/X_j \in \Pi_i} \tilde{d}_j - 1$. Thus, considering that by definition $2^{\tilde{d}_i} = d_i$,[6] the local complexity of the algorithm for these two worst cases is respectively $O(t \cdot (d_i \cdot d_{i+n})^2)$ and $O(t \cdot (d_{i+n} \cdot \prod_{j/X_j \in \Pi_i} d_j)^2)$.

Globally, any iteration of 2U is linear in the size (i.e., the longest path) of the net, and the size of the exact binarization grows of a factor at most equal to $2 \cdot \max_{i=1}^{2n} \tilde{d}_i$ with respect to the original net. The factor depends (i) on the decision-theoretic transformation that doubles the number of nodes, and on (ii) the binarization that makes of each node $X_i \in \mathbf{X}'$ a cluster of binary nodes $\tilde{X}_i$ whose size depends on the logarithm $\tilde{d}_i$ of its number of states d_i. We can approximate the global complexity by assuming that a not-too-big constant bounds both the logarithms of (i) the maximum number of states for each variable in $\mathbf{X}$, and (ii) the maximum overall number of vertices of the credal sets associated to these variables. Thus, we conclude that any iteration of GL2U is roughly linear in the size of the net.

[6] As in Sect. 4.1, the number of states for each variable in $\mathbf{X}'$ is assumed to be an integer power of two. The discussion of the general case is omitted because of lack of space and will be presented in a future work.

5 Numerical Tests

In order to test the performance of GL2U, we have chosen the *Alarm* and the *Insurance* nets, as well as some random generated nets. We work with random polytrees with 50 nodes (Polyt-50), and random multiply connected nets with 10 and 25 nodes (Multi-10 and Multi-25, respectively). For the Alarm and the Insurance nets, we use the original graph with the original number of states. For the random polytrees, we generate random graphs with 50 nodes and at most 4 categories in each variable. With random multiply connected nets, we work with 10 and 25 nodes, and 4 and 8 categories.

Table 1: Average mean square errors of LS, GL2U and BIN. Maximum number of states/vertices is indicated in the second column. Best accuracies are bolded.

		LS	GL2U	BIN
Multi-10	4 / 2	0.0189	**0.0140**	0.0181
Multi-10	8 / 2	0.0195	**0.0107**	0.0338
Multi-10	4 / 4	**0.0120**	0.0175	0.0308
Multi-10	4 / 8	**0.0027**	0.0125	0.0222
Multi-10	8 / 4	0.0234	**0.0189**	0.0693
Multi-25	4 / 2	0.0231	**0.0160**	0.0184
Multi-25	4 / 4	0.0248	**0.0204**	0.0303
Polyt-50	4 / 2	**0.0112**	0.0193	0.0289
Polyt-50	4 / 4	**0.0145**	0.0221	0.0392
Insurance	5 / 2	**0.0055**	0.0117	0.0175
Insurance	5 / 4	**0.0113**	0.0132	0.0193
Alarm	4 / 2	0.0290	**0.0190**	0.0302
Alarm	4 / 4	0.0331	**0.0239**	0.0423

We run marginal inferences using GL2U, the "rough" binarization without DTS (BIN), the approximate *local search* method (da Rocha et al., 2003) (LS) limited to 20 iterations in order to have running times similar to those of GL2U, and the exact method presented in (de Campos and Cozman, 2007). Tab. 1 shows

the mean square errors. GL2U improves, often substantially, the approximation accuracy when compared to BIN; moreover, it has accuracy similar to LS. Moreover, the running time and the amount of allocated memory for LS rapidly increases with the size of the net, that makes unfeasible a solution for large nets, which can be instead quickly updated by GL2U (see Fig. 4).[7] As far as we know other existing algorithms besides LS are at least exponential in the treewidth of the moralized graph and suffer from the same complexity issues. In fact, some comparisons have been done also with the *hill-climbing* algorithm in (Cano et al., 2007) and a behavior very similar to that in Fig. 4 has been observed. Hence, GL2U has a great expected speed up with respect to them.

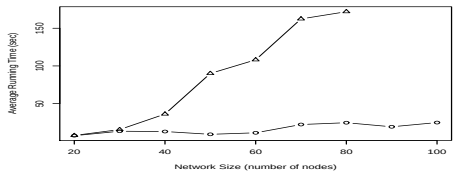


Figure 4: Average running time versus net size for LS (triangles) and GL2U (circles). LS cannot solve CNs with more than 80 nodes.

6 Conclusions

This paper has proposed a new approximate algorithm for CNs updating. This task is achieved augmenting the net by a number of nodes enumerating the vertices of the credal sets and then transforming the CN in a corresponding net over binary variables, and updating such binary CN by the loopy version of 2U. The procedure applies to any CN, without restrictions related to the topology or the number of possible states, and the only approximation is due to the loopy propagation. Empirical analysis shows that GL2U is a competitive procedure for approximate inference in CNs both in terms of accuracy and scalability. The algorithm is purely distributed and allows for simultaneous updating of all the variables in the net: these characteristics are usually not shared by optimization-based algorithms. Moreover, the computational complexity of GL2U makes it possible to solve large nets, which cannot be updated, at least with the same accuracy, by existing algorithms.

[7] A software implementation of GL2U is freely available at www.idsia.ch/∼sun/g2lu.html. The running times in Fig. 4 refer to an earlier implementation, while the last release is significantly faster.

Acknowledgments

Work partially supported by the Swiss NSF grants 200021-113820/1 and 200020-116674/1, and Hasler Foundation grant 2233.

A Proofs

Lemma 1. *Consider a CN with a single node X and vacuous $K(X) := K_{\Omega^*_X}(X)$, where $\Omega^*_X \subseteq \Omega_X$. Let $\tilde{K}(\tilde{X})$ denote the strong extension of its binarization (as in Sect. 3.1). Then:*

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

Proof. Consider a generic $\tilde{P}_*(\tilde{X}) \in \text{ext}[\tilde{K}(\tilde{X})]$, where $\tilde{X} := (\tilde{X}^0, \ldots, \tilde{X}^{\tilde{d}-1})$ with $\tilde{d}$:=$\log_2|\Omega_X|$. A corresponding mass function $P_*(X) := \tilde{P}_*(\tilde{X})$ can be therefore defined. Thus:

$$\tilde{P}_*(\tilde{x}) = \prod_{j=0}^{\tilde{d}-1} \tilde{P}_*(\tilde{x}^j|\tilde{x}^{j-1}, \ldots, \tilde{x}^0), \tag{9}$$

for each $\tilde{x} \in \Omega_{\tilde{X}}$ such that $(\tilde{x}^0, \ldots, \tilde{x}^{\tilde{d}-1}) = \tilde{x}$. For each j=$0, \ldots, \tilde{d}-1$ and each possible value of their parents, the conditional mass functions $\tilde{P}_*(\tilde{X}^j|\tilde{x}^{j-1}, \ldots, \tilde{x}^0)$ are vertices of their corresponding conditional credal sets because of Proposition 1 of (Antonucci and Zaffalon, 2008). Thus, the values of the conditional probabilities on the right-hand side of Eq. (9) are obtained by a minimization as in Eq. (5). The values to be minimized are obtained from Eq. (6), where the conditional probabilities on the right-hand side are the vertices of $K(X)$, i.e., the $m := |\Omega^*_X|$ degenerate extreme mass functions of the vacuous credal set $K_{\Omega^*_X}(X)$. This means that there is only a non-zero term in the sum in Eq. (6) and therefore each vertex of $K_{\Omega^*_X}$ produces a degenerate conditional mass function for the corresponding binary variable. Consequently, also the extreme values returned by Eq. (5) will be

degenerate. We can therefore conclude that, according to Eq. (9), also $\tilde{P}_*(\tilde{X})$ and hence $P_*(X)$ is a degenerate mass functions. Let $x_* \in \Omega_X$ be the state of X such that $P_*(x_*) = 1$. Considering Eq. (9) for $\tilde{x}_* \in \Omega_{\tilde{X}}$, we conclude that all the conditional probabilities on the right-hand side are equal to one. Considering the highest order bit, according to Eq. (6) and denoting by $P_k(X)$ a vertex of $\Omega_*(X)$, we have $\tilde{P}_*(\tilde{x}_*^{\tilde{d}-1}|\tilde{x}_*^{\tilde{d}-2}, \ldots, \tilde{x}_*^0) = P_k(x_*) = 1$, that requires $x_* \in \Omega_X^*$. Thus, $P_*(X) \in \text{ext}[K(X)]$, that implies $\text{ext}[\tilde{K}(\tilde{X})] \subseteq \text{ext}[K(X)]$, and finally $\tilde{K}(\tilde{X}) \subseteq K(X)$. On the other side, $\tilde{K}(\tilde{X}) \supseteq K(X)$ because of Th. 2 in (Antonucci et al., 2006), and hence the thesis. □

Proof of Th. 1. Given a $\tilde{P}'_*(\tilde{\mathbf{X}}') \in \text{ext}[\tilde{K}'(\tilde{\mathbf{X}}')]$, the following factorization holds:

$$\tilde{P}'_*(\tilde{\mathbf{x}}') = \prod_{i=1}^{2n} \prod_{j=0}^{\tilde{d}_i - 1} \tilde{P}'_*(\tilde{x}_i^j|\tilde{\pi}_i^j) = \prod_{i=1}^{2n} \tilde{P}'_*(\tilde{x}_i^0, \ldots, \tilde{x}_i^{\tilde{d}_i-1}|\tilde{\pi}'_i), \tag{10}$$

for each $\tilde{\mathbf{x}}' \in \Omega_{\tilde{\mathbf{X}}'}$, where the values of the other variables are consistent with $\tilde{\mathbf{x}}$, and the last equality follows from chain rule. Eq. (10) defines $P'_*(X_i|\pi'_i) := \tilde{P}'_*(\tilde{X}_i^0, \ldots, \tilde{X}_i^{\tilde{d}_i-1}|\tilde{\pi}'_i)$. As noted in Sect. (3.2), for each $i = 1, \ldots, n$ and $\pi_i \in \Omega_{\Pi_i}$, $K'(X_i|\pi'_i)$ is a credal set made of a single point. Thus, as a corollary of Th. 1 in (Antonucci et al., 2006), we have $P'_*(X_i|\pi'_i) \in \text{ext}[K'(X_i|\pi'_i)]$, being in fact the only element of this credal set. Similarly, for each $i = 1, \ldots, n$, the credal set $K'(X_{i+n}|\pi'_{i+n})$ is vacuous. Thus, regarding this credal set as a CN made of a single node, we invoke Lemma 1 and obtain from $\tilde{P}'_*(\tilde{X}_{i+n}|\tilde{\pi}'_{i+n}) \in \text{ext}[\tilde{K}'(\tilde{X}_{i+n}|\tilde{\pi}'_{i+n})]$ that $P'_*(X_{i+n}|\pi'_{i+n}) \in \text{ext}[K'(X_{i+n}|\pi'_{i+n})]$. Overall, we proved that $P'_*(\mathbf{X}')$ is a combination of local vertices of the credal sets of $\langle \mathcal{G}', \mathbf{P}'(\mathbf{X}')\rangle$. Thus, $P'_*(\mathbf{X}') \in \text{ext}[K'(\mathbf{X}')]$, from which $\text{ext}[\tilde{K}'(\tilde{\mathbf{X}}')] \subseteq \text{ext}[K'(\mathbf{X}')]$, and finally $\tilde{K}'(\tilde{\mathbf{X}}') \subseteq K'(\mathbf{X}')$. According to Lemma 1 in (Antonucci et al., 2006), $\tilde{K}'(\tilde{\mathbf{X}}') \supseteq K'(\mathbf{X}')$. Thus, $\tilde{K}'(\tilde{\mathbf{X}}') = K'(\mathbf{X}')$. Marginalizing on both the sides we get $\tilde{K}'(\tilde{\mathbf{X}}) = K'(\mathbf{X})$. But Th. 2 in (Antonucci and Zaffalon, 2008) states $K(\mathbf{X}) = K'(\mathbf{X})$, from which the thesis. □

References

A. Antonucci and M. Zaffalon. 2008. Decision-theoretic specification of credal networks: A unified language for uncertain modeling with sets of Bayesian networks. *Int. J. Approx. Reasoning*. Forthcoming.

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*, pages 120–131, Amsterdam. IOS Press.

A. Cano, M. Gómez, S. Moral, and J. Abellán. 2007. Hill-climbing and branch-and-bound algorithms for exact and approximate inference in credal networks. *International Journal of Approximate Reasoning*, 44(3):261–280.

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

J. C. da Rocha, F. G. Cozman, and C. P. de Campos. 2003. Inference in polytrees with sets of probabilities. In *Conference on Uncertainty in Artificial Intelligence*, pages 217–224, Acapulco.

C. P. de Campos and F. G. Cozman. 2005. The inferential complexity of Bayesian and credal networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1313–1318, Edinburgh.

C. P. de Campos and F. G. Cozman. 2007. Inference in credal networks through integer programming. In *Proceedings of the Fifth International Symposium on Imprecise Probability: Theories andApplications*, Prague. Action M Agency.

E. Fagiuoli and M. Zaffalon. 1998. 2U: an exact interval propagation algorithm for polytrees with binary variables. *Artificial Intelligence*, 106(1):77–107.

J. S. Ide and F. G. Cozman. 2004. IPE and L2U: Approximate algorithms for credal networks. In *Proceedings of the Second Starting AI Researcher Symposium*, pages 118–127, Amsterdam. IOS Press.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo.

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

Carmen: An open source project for probabilistic graphical models

Manuel Arias and Francisco J. Díez
Dept. Inteligencia Artificial. UNED
Juan del Rosal, 16. 28040 Madrid

Abstract

Carmen is an open-source software package for probabilistic graphical models (PGMs), which aims at being useful for different research groups and for building real-world applications. After reviewing similar projects launched in the last years, we analyze the general properties of Carmen and how it adheres to the principles of software engineering, in particular by including exhaustive documentation and systematic tests. We then describe the current state of development, i.e., which algorithms and modules have been implemented so far, and discuss the results of a preliminary analysis of its performance.

1 Introduction

Graphical probabilistic models (PGMs), such as Bayesian networks and influence diagrams, are a powerful tool for uncertain reasoning in real-world problems (Pearl, 1988). Several computer packages for building and evaluating PGMs have been developed in the last years—see K. Murphy's list at `www.cs.ubc.ca/~murphyk/Software/bnsoft.html`. In particular, many researchers, practioners, and teachers of PGMs felt the advisability of having an open source package supported by a large community of programmers. One of the main reasons is that every people or group who wished to use a PGM had to do a lot of programming from scratch. Another reason was that an open source tool would allow to compare the performance of different algorithms.

Several projects started with the objective of building such a tool. However, in our opinion, it still makes sense to try to build an open source tool for PGMs, as we discuss at length in Section 2. The tool that we present in this paper is Carmen.

The rest of the paper is structured as follows. In Section 2 we review some of the open source packages developed in the last years. Then we describe Carmen's general properties (Sec. 3), the methods and modules implemented so far (Sec. 4) and a rough premilinary comparison of Carmen's performance with three well-known tools (Sec. 5). In Section 6 we state the delivery schedule and conclude in Section 7.

In this paper we assume that the reader is familiar with PGMs. Due to the lack of space, we do not give references to standard algorithms, such as variable elimination, Hugin, lazy propagation, etc., nor for well-known computer applications, such as CVS, subversion, listserv, etc.

2 Previous work

In Table 1 we list some open source programs for PGMs.[1] In the following, we analyze four of the most successful packages: BNT, PNL, Weka, and Elvira, in order to show that, despite all these open-source projects started in the last years, it still makes sense to propose a new one that aspires to involve developers from different research groups.

[1] Some members of the *free software community* insist on the difference between "free software" and "open source software", arguing that the latter is confusing. However, for most of the people "open source" means not only that the source code is visible, but also that it can be modified and distributed; i.e., both terms are synonymous. Furthermore, we prefer "open source" because it avoids the ambiguity of "free": for instance, in K. Murphy's list, the latter does not mean "free software" but "free of charge".

	BNT	PNL	OBP	JavaBayes	BNJ	Riso	BayesLine	Weka	Elvira	Carmen
Language	Matlab	C++	Python	Java	Java	Java	Java	Java	Java	Java
License	GPL	IOSL	GPL	GPL	GPL	GPL	LGPL	GPL		LGPL
User manuals	yes	no	yes	yes	yes	yes	no	yes	yes	yes
Users list/forum	yes	no	yes	no	yes	yes	yes	yes	yes	yes
Developer manuals	no	no	no	no	no	no	no	yes	yes	yes
Developers list	yes	no	yes	no	yes	yes	yes	yes	yes	yes
Source HTML docs	no	no	no	yes	no	yes	yes	yes	yes	yes
Version control	no	no	yes	no	yes	yes	yes	yes	yes	yes
Bug tracker	no	no	yes	yes	yes	yes	yes	yes	no	yes
Start	1999	2003	2006	1996	2004	2000	2003	1993	1997	2004
Stopped	2007	2005	2007	2001	2004	2004	2003	–	–	–

Table 1: Open-source packages for PGMs. The URLs for these packages can be found in K. Murphy's list, at `www.cs.ubc.ca/~murphyk/Software/bnsoft.html`. (OBP = OpenBayes for Python. IOSL = Intel Open Source License.)

2.1 BNT and its successors

BNT (BayesNetToolbox) was built by Kevin Murphy (2001) on Matlab, a numerical matrix-oriented environment, which includes a specific programming language. The reasons for its success are the many features implemented in BNT (Bayesian networks, influence diagrams, dynamic models, learning and inference with both discrete and continuous variables...), its robustness, and the clarity of the code and its documentation.

The main limitations of BNT stem from its dependence on Matlab, which requires an expensive licence (although there is a relatively cheap student licence), is much slower than other programming languages, and has limited support for object-oriented programming.

This has motivated several attempts to build a new program on an efficient language, but none of them has been able to replace BNT. The **OpenBayes** project, initiated by R. Dybowski and K. Murphy in 2001, was abandonned some months later without even arriving at a decision about which programming language to use. In 2005, Intel Labs in Russia, with the collaboration of K. Murphy, released **PNL** (Probabilistic Networks Library), an open-source C++ library for PGMs. The project was abandonned that same year. Similarly, the project **OpenBayes for Python** was stopped before the release of version 0.2, which was announced in February 2007.

2.2 Weka

This project started in 1993 at the University of Waikato, in New Zealand, with the purpose of including several data mining methods in a single tool. The Bayesian network (BN) classifiers were implemented by Remco R. Bouckaert (2004).

The main advantages of Weka is that it has good documentation, which facilitates the development of new learning algorithms, and the possibility of empirically comparing many methods, not only those that build BNs, but also "traditional algorithms", like C4.5, nearest neighbors, etc.

2.3 gR

R is "a language and environment for statistical computing and graphics". It is an evolution of a previous language, S. gR was a subproject of R aimed at providing facilities for PGMs. The development of gR stoppped in 2003.

2.4 Elvira

Elvira (Elvira Consortium, 2002) started in 1997 as a join project of several Spanish universities. The main objectives were to foster the collaboration of the different groups working on PGMs in our country and to build a package

that be (1) a workbench for new PGM algorithms, (2) a tool for tuition, (3) a tool for building real-world applications, and (4) an open-source program. It was very successful in almost all of them. The meetings of the project were an interesting forum for the exchange of ideas. The Elvira program, whose development still continues, has served to try new methods that led to around 15 or 20 PhD theses and many papers; as a consequence, Elvira is probably the tool having more algorithms for inference and learning PGMs. As a tool for tuition, Elvira has been used by hundreds of computer science studends and postgraduate medical students, in at least 8 countries. Several applications were built or are currently under development using Elvira, most of them for medical problems, but also for other domains.

However, Elvira has not been so successful as an open-source tool: as far as we can tell, aside from the Spanish groups involved in the project, only a few Mexican researchers use it. Some European colleagues who started using it gave up because of the difficulties they encountered. In our opinion, it was a consequence of the "publish or perish" pressure: some members of Elvira argued that writing extensive documentation, reorganizing the source code, optimizing the basic data structures, improving the interface, giving support to external developers, etc., are time-consuming tasks that return little benefit.

On the other hand, an important drawback of Elvira is the lack of efficiency (see the experiments in 5), due to the fact that some of the basic data structures and algorithms do not scale up properly. Unfortunately, any attempt to optimize one class or method might cause an unpredictable number of conflicts with other classes: currently Elvira contains over 600 Java files; the biggest one occupies 390 KB and has almost 10,000 lines; the second and the third occupy around 200 KB. Finally, the program is still buggy, mainly its GUI, but debugging Elvira is not easy. Similarly, adding new functionalities is more and more a daunting task.

These are the main reasons why we decided to build a new tool, that takes profit from the many good ideas implemented in Elvira, and also from the lessons that we learnt during its development, thus trying to avoid some of the mistakes that we made.

2.5 Discussion

The end of the development of BNT and the "death" of its successors shows that it is still desirable to build an open-source program that does not depend on Matlab. Another drawback of BNT is that it has ceased to add new features—see `www.cs.ubc.ca/~murphyk/Software/BNT/whyNotSource`

Weka might be a good candidate, but in our opinion, a general-purpose package for PGMs should be developed as an independent project, rather than as an appendix of a data mining tool: even though the two fields overlap in the construction of BN classifiers, they have very different goals and needs. For instance, when building stand-alone applications, such as medical expert systems, it would be desirable to have a library that has only some inference algorithms, without the need to include all the learning methods implemented in Weka. Additionally, it is significant, in our opinion, that despite the excellent work done by R. Bouckaert, nobody in the PGM community has tried to use Weka for any purpose other than learning BNs.

Elvira has many of the features of an open source program that might be used by a large community of programmers, but unfortunately this was not one of the priorities of the project: its developers concentrated their efforts in creating new algorithms rather than in analyzing design issues (which would have made the code more efficient and easier to mantain) and writing extensive documentation (which would facilitate the task of external developers).

We have discussed in detail these packages to explain why, in spite of the excellent contributions that they have made, we think that it still makes sense to develop a new open source package. In the rest of the paper, we will try to show how the new package that we are developing might meet the needs and expectations of the PGM community.

3 Carmen's general properties

3.1 Programming language

The development language for Carmen is Java, mainly in order to allow it to run on different platforms. Since the begining we used version 1.5, which introduced new syntactical features, such as typed collections (for instance, `ArrayList<CertainClass>`) and enhanced loops, which significantly facilitate the iteration on lists.

3.2 Documentation

There are two kinds of documentation for the Carmen project: on-line comments and PDF documents. On-line documentation consists of comments included in the Java files, and works in combination with Sun's Javadoc utility,[2] which generates a set of HTML pages with many cross-references. We devoted a significant effort to carefully choosing the names of fields, methods, and variables, and to writing thorough and clear explanations.

Sun has proposed a set of tags for documenting the Java code, such as `@author`, `@param`, `@return`, etc. We have extended this collection with new tags especially intended for guaranteeing the correctness of the code. For instance, `@precondition` indicates a condition that must be fulfilled before invoking a certain method; `@postcondition` is a logical condition fulfilled after the execution of a certain method; `@paramCondition` indicates a property that the parameters must satisfy; `@invariant` refers to a logical property always satisfies by the objects of a certain class; `@frozen` means that an attribute is set by the constructor and will not be modified afterwards; and `@sideEffec`t refers to secondary effects of the the execution of a method. These tags may be useful for the verification of the source code by human programmers and in the future might be adapted to be used by automatic verification tools.

Additionally, we are generating PDF documents, which offer a general overview of Carmen and contain several UML diagrams, mainly of types class, object, sequence, and components.

[2] `http://java.sun.com/j2se/javadoc`.

3.3 Testing

We have built a test suite for each class in Carmen with JUnit.[3] After introducing a modification in Carmen, we run the battery of tests in order to detect possible bugs in the program. It would be desirable that each new package contributed by an external developer come with its JUnit test, at least for those packages thay might be used by real-world applications.

3.4 Version control and support for developers

As a version control tool, we chose subversion, which offers several advantages over CVS. Following subversion's standard, Carmen's repository is organized in three directories: trunk, branch, and tag, which facilitates the collaboration of different programmers. The utility WebSVN allows Carmen's developers to receive customized notifications of changes in subversion, via RSS. In the near future we will install a distribution list (majordomo or listserv) and a bug tracking utility (probably Trac,[4] because it was designed to integrate with subversion). Later on, we will set up a web utility for the automatic registration of users and developers.

In our opinion, these facilities are a requisite for the success of any open source project.

4 Carmen's implementation

4.1 Basic data structures

The package graph implements graph operations, such as adding nodes or links. In principle it can be used for any kind of graph. In the case of a probabilistic network, each node represents a chance variable, a decision, or a utility; in a cluster tree, each node represents a set of chance variables; in the case of a Markov transition diagram, each node might represent a state of a variable.

The package networks is specific of probabilistic graphical models. Each network has an associated graph and a set of restrictions (see Figure 1). This allows to maximun flexibility for defining new types of PGMs. For example, the

[3] `http://www.junit.org`.

[4] `http://trac.edgewall.org`.

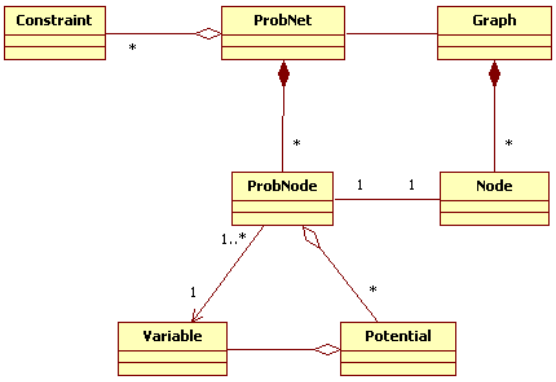


Figure 1: Main data structures for probabilistic networks.

only default restriction for a Markov network is OnlyUndirectedLinks. Both a Bayesian network and an influence diagram have the restrictions OnlyDirectedLinks and NoCycles; the former also has an OnlyChanceNodes restriction, while the latter has additional restrictions for preventing certain types of links; for instance, the children of utility nodes can only be utility nodes.

The distinction between ProbNode and Variable (cf. Fig. 1) is introduced to save memory space when a variable shared by several structures; for instance, a Bayesian network, its potentials, its moral graph (in fact, a Markov network), and a clique tree.

The package editSupport has two purposes: to allow undo/redo operations on ProbNets and to inform the listeners, i.e., the objects interested in the changes performed on a probNet. Each modification, such as adding or removing a node or a link or modifying a potential, is performed by first creating an instance of PNEdit and then passing this object to an instance of PNESupport (where PNE stands for "probabilistic network edit"), which informs the listeners, executes the "edit", and tracks it in a pushdown list to be able to undo it if necessary.

4.1.1 Edits

In general, each modification of a probabilistic network is performed by building a PNEdit. This design address three goals. First, to implement the undo/redo operations, because class PNEdit implements the Java Swing interface UndoableEdit. Second, to supervise the fulfillment of restrictions. For instance, the NoCycles restriction may register as a listener at a ProbNet to be able to veto certain additions of links. And third, to inform other classes about the changes performed to a ProbNet. For instance, an elimination heuristic might store the number of neighbors of each node, or a learning algorithms might store some scores in a cache; such information should be updated after the removal of a variable or the addition of a link. Similarly, the GUI might be interested in displaying the changes introduced by an inference algorithm (like variable elimination or arc reversal) or by a learning algorithm. This way, the possibility of adding listeners to the network offers a high degree of flexibility for fulfill these purposes and many others that future developers might imagine.

4.2 Inference

4.2.1 Purely probabilistic networks

The package inference contains methods for computing the posterior probabilities of Bayesian networks, Markov networks, and in general any kind of network that satisfies the OnlyChanceNodes restriction. The key feature of these models is that the joint probability is given by the product of a list of probabilistic potentials.

Each inference algorithm is implemented as a class that implements the Java interface EvidencePropagation, whose main methods are individualProbabilities and joinProbability. The former receives a list of variablesOfInterest and an evidenceCase, and returns a list of potentials, each one giving the posterior probability of a variable. An evidence case consists of several findings. Each finding is formed by a variable and the value that it takes. The method joinProbability receives a list of variables of interest, called query, and an evidenceCase, and returns a single potential.

Currently, we have implemented three inference algorithms for purely probabilistic networks: variable elimination, Hugin propagation, and lazy propagation. The two latter are

implemented as subclasses of ClusterPropagation, because they operate on the same structure, HuginForest. Currently Carmen offers three elimination heuristics: SimpleElimination, which chooses the node having fewer neighbors, CanoMoralElimination (Cano and Moral, 1995), and FileElimination.[5]

In the future, we will propose our students to add other heuristics and other exact and approximate algorithms, and to carry out experimental comparisons among them.

4.3 Influence diagrams

We have implemented the standard variable elimination method for influence diagrams (Jensen and Nielsen, 2007) and will later code a variable elimination method for diagrams with super-value nodes (Luque and Díez, 2004).

4.4 Learning

Carmen can learn Bayesian networks from databases using basic *search and score* techniques. The only search method implemented so far is hill climbing. The metrics implemented currently are Bayesian (which includes K2 and BDe as particular cases), cross-entropy, AIC, and MDL—see (Bouckaert, 2004; Neapolitan, 1990) for references. The file formats that Carmen can read are `dbc` (used by Elvira), `arff` (used by Weka), and Microsoft Excel.

In the future we will add other search methods, learning algorithms based on the detection of conditional independences, and learning algorithms for databases with missing values.

4.5 Markov models

One of our postgraduate students is implementing dynamic Bayesian networks, DBNs (Dean and Kanazawa, 1989), and factored Markov decision processes, MDPs (Boutilier et al., 2000), and another one will implement dynamic limited-memory influence diagrams, DLIMIDs (van Gerven et al., 2007). Later we would like to add partially observable Markov decision processes, POMDPs (Åström, 1965), which are much more difficult to solve than MDPs.

4.6 Graphical user interface (GUI)

Two undergraduate students are implementing a GUI for Carmen, whose look will be very similar to Elvira's.

5 Performance of Carmen

As a first approach to assessing the performance of Carmen, we have compared it with some well-known software tools, such as Elvira (version 0.16), GeNIE (v. 2.0), Hugin (v. 5.6), and Netica (v. 3.14).[6] We have built some test networks with a double requirement: small number of nodes and states (to make the network tractable by the demo versions of some programs) and big clusters in the clique tree, to make the measurement of time more reliable.

A solution has been the definition of networks containing $m \times n$ nodes $X_{i,j}$, m nodes R_i, and n nodes C_j. Each node R_i is a child of all the $X_{i,j}$'s (the i-th row) and each C_j is a child of all the $X_{i,j}$'s (the j-th column), with $0 \leq i \leq m-1$ and $0 \leq j \leq n-1$. The size of the largest clique is roughly $m \times n$, which means that the complexity of a network grows extremely fast with the number of nodes.

We have made some experiments with a 6×6 network by introducing evidence on all the R and C nodes. Both GeNIE and Netica needed around 3-4 seconds to compile the network and 1.5 seconds to propagate evidence, i.e., to compute the posterior probabilities of all the X nodes. Hugin needed around 10 seconds to compile the network and the same amount of time to propagate evidence; these times were the same for the four triangulation algorithms offered by that version of Hugin. Carmen, in turn, needed 0.42 seconds to compile the network and 16.0 to propagate evidence. Given that GeNIE and Netica are all implemented in C or C++, it is not surprising that they are around 10 times

[5] FileElimination is not properly a heuristic method, because it reads the list of variables from a file. It is used for forcing Carmen to use a certain elimination order, for sake of comparison with other software tools (see Sec. 5).

[6] See `www2.sis.pitt.edu/~genie`, `www.hugin.com`, and `www.norsys.com`.

faster than Carmen, which is implemented in Java.

Elvira ran out of memory for the 6×6 network; when both tools were compared on a 5×5 network, Carmen was over 100 times faster than Elvira.

The fact that in our experiments Hugin was slower than GeNIE and Netica might due to the fact that we used an old version of that program, which seems to be based on non-efficient triangulations. In fact, when analyzing Hugin's log files, we saw that the biggest clique for that network contained 24 variables, while Carmen, which used the heuristic method by Cano and Moral, built a tree whose biggest clique contains only 22 variables. When we forced Carmen to use the same elimination ordering as Hugin and, consequently, to propagate evidence on a tree containing the same cliques, Carmen needed 50.4 seconds, i.e., it was five times slower than Hugin with the same triangulation and it was three times slower than Carmen itself with the tree built with the Cano-Moral heuristic.

However, the fact that Carmen is slower when propagating evidence can be compensated by the fact that it is able to compile the network around 7 to 10 times faster than GeNIE and Netica. This means that, instead of always using the same clique tree, we can speed up the propagation of evidence by pruning the *barren nodes* (i.e., nodes that are neither variables of interest nor parents of evidence nodes) before compiling the network, which in general speeds up significantly the propagation of evidence: the time spent in compiling the pruned network is negligible compared to the time saved in the phase of inference.

In any case, we insist, these are only very preliminary results. It is necessary to perform further experiments with diffent kinds of networks, such as as those in the *Bayesian Network Repository*,[7], comparing Carmen with other software tools.

[7] `www.cs.huji.ac.il/labs/compbio/Repository`.

5.1 Discussion

In the implementation of Carmen we have used several well-known software design patterns (Gamma et al., 2005): the undo/redo operations are based on the Command pattern, listeners an example of Observer, etc. In the same way, the GUI will be based on the arquitectural pattern MVC (Model-View-Controler). Some of these patterns have been combined and adapted to our particular needs. In a future paper we will analyze in detail how we are applying the principles and methods of software engineering in an attempt to make Carmen as efficient, robust, clearly organized, and extensible as possible.

6 Release schedule

We intend to release a beta version of Carmen before the end of 2008. As an advance, some preliminary information can be found at `http://www.cisiad.uned.es/carmen`. It is possible to browse the JavaDoc pages, linked to the source code, at `http://www.cisiad.uned.es/carmen/javadoc`. After receiving the feedback from the PGM community (hopefully), we will later release the first stable version of Carmen.

7 Conclusion

In Section 2 we showed that, even thought it may seem that there are many open source packages for PGMs, only Weka and Elvira are currently active, and we argued that it still makes sense to offer a new package that might be useful for researchers of different groups and for building real-world applications. For this reason we decided to build Carmen, a project in which we are trying to adhere to the principles of software engineering in order to make our package robust, efficient, scalable, and extensible. A particular effort has been devoted to clearly documenting the source code, by means of tools such as Javadoc and UML, not only to facilitate the work of the programmers that will use Carmen, but also as a requisite to make the software robust to future additions and changes. We have also developed an extensive battery of

tests (in JUnit) for checking the stability of Carmen under new modifications.

We have already implemented algorithms for inference in Bayesian networks and influence diagrams with discrete variables, as well as the standard "search and score" learning algorithms. Other learning methods, several types Markovian decision models, and a GUI are under development.

A preliminary evaluation of Carmen's performance seems to indicate that it is efficient enough (when compared to commercial tools) to be used in real-world applications.

We would like to present Carmen to the PGM community at the PGM-08 conference, in Hirtshals, Denmark, to attrack the interest of other researchers that might be willing to contribute to this project.

Acknoledgements

José E. Mendoza and Alberto M. Ruiz are implementing Carmen's GUI, Jorge Fernández is implementing Markov decision processes, and Jesús Oliva is implemeting some learning algorithms, all under the supervision of the authors of this paper. Manuel Luque has helped to set up a web server for Carmen.

We thank all the members of the Elvira project for all that they have taught us.

This work has been supported by the Spanish Ministry of Education and Science, under grant TIN-2006-11152.

References

[Åström1965] K. J. Åström. 1965. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205.

[Bouckaert2004] R. R. Bouckaert. 2004. Bayesian networks in Weka. Technical Report 14/2004, Computer Science Department, University of Waikato, New Zealand.

[Boutilier et al.2000] C. Boutilier, R. Dearden, and M. Goldszmidt. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107.

[Cano and Moral1995] A. Cano and S. Moral. 1995. Heuristic algorithms for the triangulation of graphs. In B. Bouchon-Meunie, R. R. Yager, and I. A. Zadeh, editors, *Advances in Intelligent Computing (IPMU-94)*, pages 98–107. Springer-Verlag, Berlin.

[Dean and Kanazawa1989] T. Dean and K. Kanazawa. 1989. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150.

[Elvira Consortium2002] 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.

[Gamma et al.2005] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. 2005. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Boston, MA.

[Jensen and Nielsen2007] F. V. Jensen and T. D. Nielsen. 2007. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, second edition.

[Luque and Díez2004] M. Luque and F. J. Díez. 2004. Variable elimination for influence diagrams with super-value nodes. In P. Lucas, editor, *Proceedings of the Second European Workshop on Probabilistic Graphical Models*, pages 145–152.

[Murphy2001] K. Murphy. 2001. The Bayes net toolbox for Matlab. *Computing Science and Statistics*, 33:1–20.

[Neapolitan1990] R. E. Neapolitan. 1990. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. Wiley-Interscience, New York.

[Pearl1988] J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

[van Gerven et al.2007] M. A. J. van Gerven, F. J. Díez, B. G. Taal, and P. J. F. Lucas. 2007. Selecting treatment strategies with dynamic limited-memory influence diagrams. *Artificial Intelligence in Medicine*, 40:171–186.

Bayesian Networks: the Parental Synergy

Janneke H. Bolt
Department of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

In a Bayesian network, for any node its conditional probabilities given all possible combinations of values for its parent nodes are specified. In this paper a new notion, the *parental synergy*, is introduced which can be computed from these probabilities. This paper also conjectures a general expression for the error which is found in the marginal prior probabilities computed for a node when the parents of this node are assumed to be independent. The parental synergy is an important factor of this expression; it determines to what extend the actual dependency between the parent nodes can affect the computed probabilities. This role in the expression of the prior convergence error indicates that the parental synergy is a fundamental feature of a Bayesian network.

1 Introduction

A Bayesian network is a concise representation of a joint probability distribution over a set of stochastic variables, consisting of a directed acyclic graph and a set of conditional probability distributions (Pearl, 1988). The nodes of the graph represent the variables of the distribution. From a Bayesian network, in theory, any probability of the represented distribution can be inferred. Inference, however, is NP-hard in general (Cooper, 1990) and may be infeasible for large, densely connected networks. For those networks, approximate algorithms have been designed. A widely used algorithm for approximate inference with a Bayesian network is the loopy-propagation algorithm (Pearl, 1988).

In Bolt and van der Gaag (2004), we studied the performance of the loopy-propagation algorithm from a theoretical point of view. We observed that in a network in its prior state, a prior convergence error may arise in the marginal probabilities computed for a node with two or more incoming arcs and noted that such an error may arise because the algorithm assumes the parents of a node to be independent, while, in fact, they may be dependent. Thereafter, for binary networks, we derived an expression for the prior convergence error found in a node with two incoming arcs. This expression is composed of some factors that capture the degree of dependency between the parents of this node, and of a weighting factor w that determines to what extent this degree of dependency can contribute to the prior convergence error. The factor w is composed of the conditional probabilities specified for the node.

In this paper, the notion of parental synergy is introduced. This notion is a generalisation of the factor w and can be computed for each node, irrespective its number of parents and irrespective of the cardinality of the involved nodes. Thereafter, the expression for the prior convergence error is generalised to nodes with an arbitrary number of parents and to network with nodes of arbitrary cardinality. In this generalised expression, the parental synergy fulfils the role of weighting factor. The role of the parental synergy in the expression of the prior convergence error indicates that it captures a fundamental feature of the probability landscape in a Bayesian network.

More details about the research described in this paper can be found in Bolt (2008).

2 General Preliminaries

A *Bayesian network* is a model of a joint probability distribution Pr over a set of stochas-

tic 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:

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

Figure 1 depicts the graph of an example Bayesian network. The network includes a node C with n parents $A^1, \ldots, A^n$, $n \geq 0$. The nodes $A^1, \ldots, A^n$ in turn have a common parent D. For $n = 0$ and $n = 1$, no loop is included in the network. For $n = 2$, the graph consists of a simple loop. For $n > 2$, the graph consists of a compound loop; for $n = 3$, this compound loop will be termed a double loop.

For a Bayesian network with a graph as depicted in Figure 1, the marginal probability $\Pr(c_i)$ equals:

$$\Pr(c_i) = \sum_{\mathbf{A}} \Pr(c_i \mid \mathbf{A}) \cdot \Pr(\mathbf{A})$$

Wrongfully assuming independence of the parents $A^1, \ldots, A^n$ would give the approximation $\widetilde{\Pr}(c_i)$:

$$\widetilde{\Pr}(c_i) = \sum_{\mathbf{A}} \Pr(c_i \mid \mathbf{A}) \cdot \Pr(A^1) \cdot \ldots \cdot \Pr(A^n)$$

In the loopy-propagation algorithm (Pearl, 1988), a widely used algorithm for approximate inference, indeed the parents of a node are always considered to be independent. With this algorithm, in the network from Figure 1 for node C the probabilities $\widetilde{\Pr}(c_i)$ would be yielded.

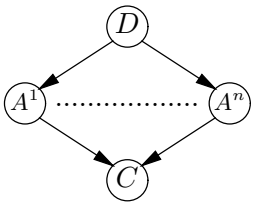


Figure 1: An example graph of a Bayesian network with a node C with the dependent parents $A^1, \ldots, A^n$.

In Bolt and van der Gaag (2004), we termed the error which arises in the marginal prior probabilities computed for a child node under assumption of independence of its parent nodes, a prior convergence error. Moreover, we analysed the prior convergence error found in a binary network with a graph consisting of a node C with just the parents A and B with the common parent D, as the network depicted in Figure 2.

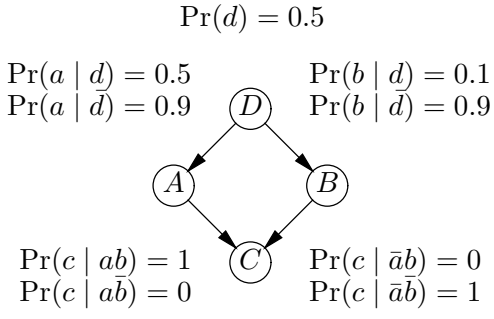


Figure 2: An example Bayesian network with a node C with the dependent parents A and B.

For the prior convergence error $v_i = \Pr(c_i) - \widetilde{\Pr}(c_i)$ in such a network the following expression was found:

$$v_i = l \cdot m \cdot n \cdot w$$

[1] Variables will be denoted by upper-case letters (A, A^i), 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. Given binary variables, $A = a_1$ is often written as a and $A = a_2$ is often written as $\bar{a}$.

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

where

$$\begin{aligned} l &= \Pr(d) - \Pr(d)^2 \\ m &= \Pr(a \mid d) - \Pr(a \mid \bar{d}) \\ n &= \Pr(b \mid d) - \Pr(b \mid \bar{d}) \\ w &= \Pr(c_i \mid ab) - \Pr(c_i \mid a\bar{b}) \\ &\quad - \Pr(c_i \mid \bar{a}b) + \Pr(c_i \mid \bar{a}\bar{b}) \end{aligned}$$

The factors were illustrated graphically with Figure 3. The line segment in this figure captures the exact probability $\Pr(c)$ as a function of $\Pr(d)$, given the conditional probabilities for the nodes A, B and C from Figure 2. $\Pr(d)$ itself is not indicated in the figure, note however, that each particular $\Pr(d)$ has a corresponding $\Pr(a)$ and $\Pr(b)$. The end points of the line segment, for example, are found at $\Pr(d) = 1$ with the corresponding $\Pr(a) = 0.5$ and $\Pr(b) = 0.1$ and at $\Pr(d) = 0$ with the corresponding $\Pr(a) = 0.9$ and $\Pr(b) = 0.9$. The surface captures $\widetilde{\Pr}(c)$ as a function of $\Pr(a)$ and $\Pr(b)$, given the conditional probabilities for node C. The convergence error equals the distance between the point on the line segment that matches the probability $\Pr(d)$ from the network and its orthogonal projection on the surface. For $\Pr(d) = 0.5$ the difference between $\Pr(c)$ and $\widetilde{\Pr}(c)$ is indicated by the vertical dotted line segment and equals $0.66 - 0.5 = 0.16$. The factor l now reflects the location of the point with the exact probability on the line segment and the factors m and n reflect the location of the line segment. The factor w, to conclude, reflects the curvature of the surface with the approximate probabilities. This curvature is determined by the change of the influence of one of the parent nodes on C occasioned by the change of the value of the other parent node. We argued that the factors l, m and n capture the degree of dependency between the parent nodes A and B and that the factor w acts as a weighting factor, determining to what extent the dependence between A and B can affect the computed probabilities.

The factor w, as used in the expression for the convergence error above, only applies to nodes with two binary parents. In the next Section, this factor is extended to a general notion which will be called the *parental synergy*. Subsequently, in Sections 4 and 5, the expression of the prior convergence error is generalised to a node C with an arbitrary number of dependent parents and with parents of arbitrary cardinality. The parental synergy is an important factor of these expressions and, analogous to the factor w, has the function of weighting factor.

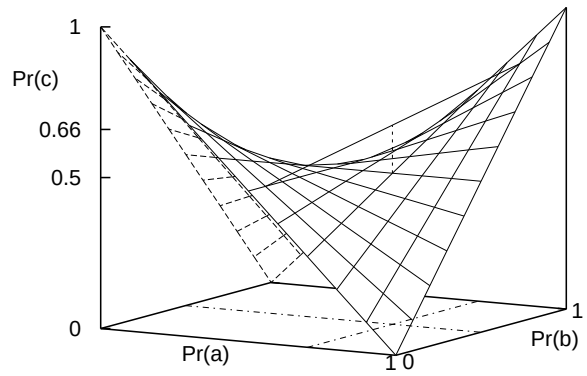


Figure 3: The line segment capturing $\Pr(c)$ and the surface capturing $\widetilde{\Pr}(c)$, for the example network from Figure 2.

3 The Parental Synergy

Before formally defining the parental synergy, the indicator function δ on the joint value assignments $a^1_{i1}, \ldots, a^n_{in}$ to a set of variables $A^1, \ldots, A^n$, $n \geq 0$, given a specific assignment $a^1_{s1}, \ldots, a^n_{sn}$ to these variables is introduced:

$$\delta(a^1_{i1}, \ldots, a^n_{in} \mid a^1_{s1}, \ldots, a^n_{sn}) = \begin{cases} 1 & \text{if } \sum_{k=1,\ldots,n} a^k_{ik} \neq a^k_{sk} \text{ is even} \\ -1 & \text{if } \sum_{k=1,\ldots,n} a^k_{ik} \neq a^k_{sk} \text{ is odd} \end{cases}$$

where true $\equiv$ 1 and false $\equiv$ 0. The indicator function compares the joint value assignment $a^1_{i1}, \ldots, a^n_{in}$ with the joint assignment $a^1_{s1}, \ldots, a^n_{sn}$, and counts the number of differences: the assignment $a^1_{i1}, \ldots, a^n_{in}$ is mapped to the value 1 if the number of differences is even and is mapped to -1 if the number of differences is odd. For the binary variables A and B,

for example, $\delta(ab \mid ab) = 1$, $\delta(a\bar{b} \mid ab) = -1$, $\delta(\bar{a}b \mid ab) = -1$ and $\delta(\bar{a}\bar{b} \mid ab) = 1$.

Building upon the indicator function δ, the notion of parental synergy is defined as follows:

Definition 3.1. Let $\mathbf{B}$ be a Bayesian network, representing a joint probability distribution Pr over a set of variables $\mathbf{V}$. Let $\mathbf{A} = \{A^1, \ldots, A^n\} \subseteq \mathbf{V}, n \geq 0$, and let $C \in \mathbf{V}$ such that C is a child of all variables in the set $\mathbf{A}$, that is, $A^j \rightarrow C, j = 1, \ldots, n$. Let $\mathbf{a}$ be a joint value assignment to $\mathbf{A}$ and let c_i be a value of C. Furthermore, let $\mathbf{X} \subseteq \rho(C) \backslash \mathbf{A}$, where $\rho(C)$ denotes the parents of C, and let $\mathbf{x}$ be a value assignment to $\mathbf{X}$. The *parental synergy* of $\mathbf{a}$ with respect to c_i given $\mathbf{X} = \mathbf{x}$, denoted as $Y^\star_{\mathbf{x}}(\mathbf{a}, c_i)$, is

$$Y^\star_{\mathbf{x}}(\mathbf{a}, c_i) = \sum_{\mathbf{A}} \delta(\mathbf{A} \mid \mathbf{a}) \cdot \Pr(c_i \mid \mathbf{A}\mathbf{x})$$

□

Given an an empty value assignment to the nodes $\mathbf{X}$, the parental synergy is denoted by $Y^\star(\mathbf{a}, c_i)$.

Example 3.2. Consider an arbitrary-valued node C with the two ternary parents A and B; the conditional probabilities for the value c_i of C given A and B, are listed in Table 1. The parental synergy $Y^\star(a_2b_2, c_i)$ of a_2 and b_2 with respect to c_i, for example, is computed from $\Pr(c_i \mid a_1b_1) - \Pr(c_i \mid a_1b_2) + \Pr(c_i \mid a_1b_3) - \Pr(c_i \mid a_2b_1) + \Pr(c_i \mid a_2b_2) - \Pr(c_i \mid a_2b_3) + \Pr(c_i \mid a_3b_1) - \Pr(c_i \mid a_3b_2) + \Pr(c_i \mid a_3b_3) = 2.0$. Table 2 lists all parental synergies $Y^\star(a_jb_k, c_i), j, k = 1, 2, 3$. □

Table 1: The conditional probabilities $\Pr(c_i \mid AB)$ for a variable C with the ternary parents A and B.

$\Pr(c_i \mid AB)$	a_1	a_2	a_3
b_1	0.7	0.2	0.3
b_2	0.2	1.0	0.8
b_3	0.4	0.1	0.9

Table 2: The parental synergies matching the conditional probabilities $\Pr(c_i \mid AB)$ from Table 1.

$Y^\star(AB, c_i)$	a_1	a_2	a_3
b_1	2.4	0.4	−0.6
b_2	−1.2	2.0	−0.2
b_3	0.8	−0.4	1.4

From the definition of parental synergy, it is readily seen that for a binary parent A^k of C, we have that $Y^\star_{\mathbf{x}}(\mathbf{a}a^k, c_i) = -Y^\star_{\mathbf{x}}(\mathbf{a}\bar{a}^k, c_i)$ for any value assignments $\mathbf{a}$ and $\mathbf{x}$. For a node C with binary parents only, therefore, for a given $\mathbf{x}$ the parental synergies with respect to some value c_i can only differ in sign. From the definition it further follows that given a binary parent A^k of C, with the values a^k_m and a^k_n, that $Y^\star_{\mathbf{x}}(\mathbf{a}a^k_m, c_i) = Y^\star_{\mathbf{x}a^k_m}(\mathbf{a}, c_i) - Y^\star_{\mathbf{x}a^k_n}(\mathbf{a}, c_i)$.

Example 3.3. Consider an arbitrary-valued node C with the ternary parent A and the binary parent B; the conditional probabilities for the value c_i of C given A and B, are listed in Table 3; the matching parental synergies are listed in Table 4. It is easily verified that $Y^\star(Ab, c_i)$ equals $-Y^\star(A\bar{b}, c_i)$ for all possible values of A. Furthermore from, for example, $Y^\star(a_1b, c_i) = (0.8 - 0.3 - 0.5) - (0 - 0.4 - 0.9) = 1.3$, $Y^\star_b(a_1, c_i) = 0.8 - 0.3 - 0.5 = 0$ and $Y^\star_{\bar{b}}(a_1, c_i) = 0 - 0.4 - 0.9 = -1.3$, it is readily verified that $Y^\star(a_1b, c_i) = Y^\star_b(a_1, c_i) - Y^\star_{\bar{b}}(a_1, c_i)$. □

Table 3: The conditional probabilities $\Pr(c_i \mid AB)$ for a node C with the ternary parent A and the binary parent B.

$\Pr(c_i \mid AB)$	a_1	a_2	a_3
b	0.8	0.3	0.5
$\bar{b}$	0.0	0.4	0.9

For a node with only binary parents, the parental synergies can be thought of as a measure of the feasible changes in its probability landscape, given a change in the value of one of its parents. This is explained in more detail below. When no parents are involved, the parental

Table 4: The parental synergies matching the conditional probabilities $\Pr(c_i \mid AB)$ from Table 3.

$Y^\star(AB, c_i)$	a_1	a_2	a_3
b	1.3	−0.5	−1.1
$\bar{b}$	−1.3	0.5	1.1

synergy with respect to a value c_i of a node C equals zero, reflecting that no change is possible. For a node C with a parent A, the parental synergy of, for example, a with respect to c_i equals $\Pr(c_i \mid a) - \Pr(c_i \mid \bar{a})$ and thus gives the feasible change in the probability of c_i, given a change of the value of A from $\bar{a}$ to a. For a node C with parents A and B, the parental synergy of, for example, ab with respect to c_i equals $\big(\Pr(c_i \mid ab) - \Pr(c_i \mid \bar{a}b)\big) - \big(\Pr(c_i \mid a\bar{b}) - \Pr(c_i) \mid \bar{a}\bar{b})\big)$[3]. It thus gives the difference between the feasible change of the probability of c_i given a change of the value of A from $\bar{a}$ to a when the value of B is b and when the value of B is $\bar{b}$[4]. And so on. Note that the number of parent nodes involved in the computation of the parental synergy can be considered to determine a kind of 'dimensionality' of the synergy. Note furthermore that for multiple-valued variables, the interpretation of the parental synergy as a measure of feasible change does not hold straightforwardly. Consider, for example, a three valued parent A with a child C. Given, for example, the conditional probabilities $\Pr(c \mid a_1) = 0.7$, $\Pr(c \mid a_2) = 0.7$ and $\Pr(c \mid a_3) = 0.7$, the parental synergy of a_i with respect to c equals $0.7 - 0.7 - 0.7 = -0.7$, whereas no change in the probability of c can be occasioned by a change in the value of A. Note, to conclude, that the parental synergy is related to the concepts of qualitative influence and additive synergy as defined for qualitative probabilistic networks by Wellman (1990). Most obviously, in a binary network, given a node C with a single parent A, the sign of the qualitative influence between A and C is computed from $\Pr(c \mid a) - \Pr(c \mid \bar{a})$, which equals $Y^\star_{\mathbf{x}}(a, c)$; given a node C with just the parents A and B the sign of the additive synergy of A and B with respect to C is computed from $\Pr(c \mid ab) - \Pr(c \mid a\bar{b}) - \Pr(c \mid \bar{a}b) + \Pr(c \mid \bar{a}\bar{b})$, which equals $Y^\star_{\mathbf{x}}(ab, c)$.

[3]Or equally $\big(\Pr(c_i \mid ab) - \Pr(c_i \mid a\bar{b})\big) - \big(\Pr(c_i \mid \bar{a}b) - \Pr(c_i \mid \bar{a}\bar{b})\big)$.

[4]Or equally: it gives the difference between the feasible change of the probability of c_i given a change of the value of B from $\bar{b}$ to b when the value of A is a and when the value of A is $\bar{a}$

4 The Convergence Error given Binary Nodes

In section 2 an expression for the prior convergence error found in the marginal prior probabilities of a node C with the two binary parent nodes A and B with a common binary parent D was given. In Section 4.1 an alternative for this expression is stated. This alternative expression is more apt for generalisation. It will be extended to apply to convergence nodes with more than two binary parents in Section 4.2 and it will be extended to multiple valued nodes in Section 5.

4.1 Two Parent Nodes; an Alternative Expression

The expression for the prior convergence error from Section 2 can also be written as

$$v_i = (s - t) \cdot w$$

where w is as before and

$$s = \sum_D \Pr(a \mid D) \cdot \Pr(b \mid D) \cdot \Pr(D)$$

$$t = \left(\sum_D \Pr(a \mid D) \cdot \Pr(D)\right) \cdot \left(\sum_D \Pr(b \mid D) \cdot \Pr(D)\right)$$

The degree of dependency between the nodes A and B now is captured by $s - t$ instead of by $l \cdot m \cdot n$. Note that the term s equals $\Pr(ab)$ and that the term t equals $\Pr(a) \cdot \Pr(b)$. Note furthermore that $w = \Pr(c_i \mid ab) - \Pr(c_i \mid a\bar{b}) - \Pr(c_i \mid \bar{a}b) + \Pr(c_i \mid \bar{a}\bar{b})$ equals $Y^\star(ab, c_i)$.

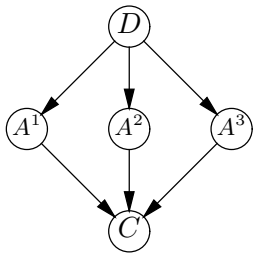


Figure 4: A node C with the dependent parents A^1, A^2 and A^3.

4.2 Multiple Parent Nodes

Consider the network from Figure 4. For three parent nodes, the expression for the prior convergence error v_i is found by subtracting the approximate probability

$$\widetilde{\Pr}(c_i) = \sum_{A^1,A^2,A^3} \Pr(c_i \mid A^1A^2A^3)\cdot \Pr(A^1)\cdot\Pr(A^2)\cdot\Pr(A^3)$$

from the exact probability

$$\Pr(c_i) = \sum_{A^1,A^2,A^3,D} \Pr(c_i \mid A^1A^2A^3)\cdot \Pr(A^1 \mid D)\cdot\Pr(A^2 \mid D)\cdot\Pr(A^3 \mid D)\cdot\Pr(D)$$

and manipulating the resulting terms. This results in the following expression

$$\begin{aligned} v_i \;=\; & (s_{a^1a^2a^3} - t_{a^1a^2a^3})\cdot w + \\ & (s_{a^2a^3} - t_{a^2a^3})\cdot w_{\bar{a}^1}(a^2a^3) + \\ & (s_{a^1a^3} - t_{a^1a^3})\cdot w_{\bar{a}^2}(a^1a^3) + \\ & (s_{a^1a^2} - t_{a^1a^2})\cdot w_{\bar{a}^3}(a^1a^2) \end{aligned}$$

where

$$\begin{aligned} s_{a^1a^2a^3} &= \sum_D \prod_{i=1,2,3} \Pr(a^i \mid D)\cdot\Pr(D) \\ t_{a^1a^2a^3} &= \prod_{i=1,2,3} \sum_D \Pr(a^i \mid D)\cdot\Pr(D) \\ w &= Y^\star(a^1a^2a^3, c_i) \end{aligned}$$

$$\begin{aligned} s_{a^ma^n} &= \sum_D \Pr(a^m \mid D)\cdot\Pr(a^n \mid D)\cdot\Pr(D) \\ t_{a^ma^n} &= \left(\sum_D \Pr(a^m \mid D)\cdot\Pr(D)\right)\cdot \left(\sum_D \Pr(a^n \mid D)\cdot\Pr(D)\right) \\ w_{\bar{a}^l}(a^ma^n) &= Y^\star_{\bar{a}^l}(a^ma^n, c_i) \end{aligned}$$

The convergence error is composed of the term $(s_{a^1a^2a^3} - t_{a^1a^2a^3})\cdot w$ which pertains to the entire double loop, and the three terms $(s_{a^ma^n} - t_{a^ma^n})\cdot w_{\bar{a}^l}(a^ma^n)$ which pertain to the three simple loops that are included within the double loop. Note that $s_{a^1a^2a^3}$ equals $\Pr(a^1a^2a^3)$; $t_{a^1a^2a^3}$ equals $\Pr(a^1)\cdot\Pr(a^2)\cdot\Pr(a^3)$; $s_{a^ma^n}$ equals $\Pr(a^ma^n)$ and $t_{a^ma^n}$ equals $\Pr(a^m)\cdot\Pr(a^m)$.

Now consider a convergence node C with the binary parents $A^1,\ldots,A^n$ and the common parent D of $A^1,\ldots,A^n$. It is posed as a conjecture that the following expression captures the prior convergence error v_i for the value c_i of C:

$$v_i \;=\; \sum_{\mathbf{m}} (s_{\mathbf{a^m}} - t_{\mathbf{a^m}})\cdot w_{\bar{a}^1\ldots\bar{a}^n\backslash\mathbf{a^m}}(\mathbf{a^m})$$

where

$$\begin{aligned} &\mathbf{m} \in \mathcal{P}(\{1,\ldots,n\}) \\ &\mathbf{a^m} = a^x\ldots a^y \quad \text{for} \quad \mathbf{m} = \{x,\ldots,y\} \\ &s_{\mathbf{a^m}} = \sum_D \prod_{i\in\mathbf{m}} \Pr(a^i \mid D)\cdot\Pr(D) \\ &t_{\mathbf{a^m}} = \prod_{i\in\mathbf{m}} \sum_D \Pr(a^i \mid D)\cdot\Pr(D) \\ &w_{\bar{a}^1\ldots\bar{a}^n\backslash\mathbf{a^m}}(\mathbf{a^m}) = Y^\star_{\bar{a}^1\ldots\bar{a}^n\backslash\mathbf{a^m}}(\mathbf{a^m}, c_i) \end{aligned}$$

in which $\bar{a}^1\ldots\bar{a}^n\backslash\mathbf{a^m}$ denotes the value assignment 'False' to the nodes included in the set $\{A^1,\ldots,A^n\}\backslash\{A^x,\ldots,A^y\}$. This expression is a straightforward generalising of the expression for the prior convergence error given three parent nodes. Note that, analogous to before, the term $s_{\mathbf{a^m}}$ equals $\Pr(a^x\ldots a^y)$ and the term $t_{\mathbf{a^m}}$ equals $\Pr(a^x)\cdot\ldots\cdot\Pr(a^y)$. Further note that

the expression includes terms for all possible loops included in the compound loop. The term with $\mathbf{m} = 1, \ldots, n$, pertains to the entire compound loop. With $|\mathbf{m}| = n - 1$, the n compound loops with a single incoming arc of C deleted are considered, and so on. Note also that, if the number of elements of $\mathbf{m}$ is smaller than two, just one parent or no parents are left; the term $s_{\mathbf{a}^\mathbf{m}}$ then equals the term $t_{\mathbf{a}^\mathbf{m}}$ and $(s_{\mathbf{a}^\mathbf{m}} - t_{\mathbf{a}^\mathbf{m}}) \cdot w_{\bar{a}^1 \ldots \bar{a}^n \setminus \mathbf{a}^\mathbf{m}}(\mathbf{a}^\mathbf{m})$ equals zero.

5 The Convergence Error given Multiple-valued Nodes

In the generalisation of the expression of the prior convergence error to multiple-valued nodes, a preliminary observation is that the expressions for this error from Section 4 involve just a single value c_i of the convergence node and therefore are valid for multiple-valued convergence nodes as well. Furthermore is observed that these expressions also provide for a multiple-valued node D. In this section expressions for the convergence error given parent nodes with an arbitrary number of values are proposed.

5.1 Two Parent Nodes

Consider a Bayesian network with a graph as the graph of network from Figure 2. It is posed as a conjecture that the following expression captures the prior convergence error for C.

$$v_i = \sum_{A,B} (s_{AB} - t_{AB}) \cdot w(AB)/4$$

where

$$s_{AB} = \sum_{D} \Pr(A \mid D) \cdot \Pr(B \mid D) \cdot \Pr(D)$$

$$t_{AB} = \left(\sum_{D} \Pr(A \mid D) \cdot \Pr(D) \right) \cdot \left(\sum_{D} \Pr(A \mid D) \cdot \Pr(D) \right)$$

$$w(AB) = Y^\star(AB, c_i)$$

This conjecture was supported by the fact that for several example networks the expression indeed yielded the prior convergence error.

Note that, analogous to the binary case, the term s_{AB} equals $\Pr(AB)$ and the term t_{AB} equals $\Pr(A) \cdot \Pr(B)$. In contrast with the binary case, now all different value combinations of the nodes A and B are considered. The impact of the dependency between a specific combination of values for the nodes A and B on the convergence error is determined by the parental synergy of this combination with respect to the value c_i of node C. Further note that the expression for the convergence error now includes a division by a constant. This constant equals 2^n, where n is the number of loop parents of the convergence node.

5.2 Multiple Parent Nodes

In Section 4.2, the expression for the convergence error was extended to convergence nodes with an arbitrary number of binary parent nodes and in Section 5.1 the expression was extended to convergence nodes with two multiple valued parent nodes. Now, it is posed as a conjecture, that these expressions combine into the following general expression for the prior convergence error. Given a network with a graph, consisting of a convergence node C with the parents $A^1, \ldots, A^n$ and the common parent D of $A^1, \ldots, A^n$, as the graph depicted in Figure 1, the convergence error equals

$$v_i = \sum_{\mathbf{m}} \Big[\sum_{\mathbf{A}^\mathbf{m}} \Big((s_{\mathbf{A}^\mathbf{m}} - t_{\mathbf{A}^\mathbf{m}}) \cdot \sum_{A^1, \ldots, A^n \setminus \mathbf{A}^\mathbf{m}} w_{A^1, \ldots, A^n \setminus \mathbf{A}^\mathbf{m}}(\mathbf{A}^\mathbf{m}) \Big) \Big] / 2^n$$

where

$$\mathbf{m} \in \mathcal{P}(\{1, \ldots, n\})$$

$$\mathbf{A}^\mathbf{m} = A^x, \ldots, A^y, \ \mathbf{m} = \{x, \ldots, y\}$$

$$s_{\mathbf{A}^\mathbf{m}} = \sum_{D} \prod_{i \in \mathbf{m}} \Pr(A^i \mid D) \cdot \Pr(D)$$

$$t_{\mathbf{A}^\mathbf{m}} = \prod_{i \in \mathbf{m}} \sum_{D} \Pr(A^i \mid D) \cdot \Pr(D)$$

$$w_{A^1 \ldots A^n \setminus \mathbf{A}^\mathbf{m}}(\mathbf{A}^\mathbf{m}) = Y^\star_{A^1 \ldots A^n \setminus \mathbf{A}^\mathbf{m}}(\mathbf{A}^\mathbf{m}, c_i)$$

Again, this conjecture was supported by the fact that for several example networks the expression indeed yielded the prior convergence error.

Note that, as before, the term $s_{\mathbf{A}^\mathbf{m}}$ equals $\Pr(A^x \ldots A^y)$ and the term $t_{\mathbf{A}^\mathbf{m}}$ equals $\Pr(A^x) \cdot \ldots \cdot \Pr(A^y)$. As in the binary case given multiple parent nodes, all combinations of parent nodes are considered, now however, for each combination of parent nodes also all combinations of value assignments to these parent nodes have to be taken into account. Again, if the number of elements of $\mathbf{m}$ is smaller than two, that is, if just one parent or zero parents are considered, then the term $s_{\mathbf{A}^\mathbf{m}}$ equals the term and $t_{\mathbf{A}^\mathbf{m}}$ and thus $\sum_{\mathbf{A}^\mathbf{m}} \Big((s_{\mathbf{A}^\mathbf{m}} - t_{\mathbf{A}^\mathbf{m}}) \cdot \sum_{A^1,\ldots,A^n \setminus \mathbf{A}^\mathbf{m}} w_{A^1,\ldots,A^n \setminus \mathbf{A}^\mathbf{m}}(\mathbf{A}^\mathbf{m}) \Big)$ equals zero. The general expression, shows that, also in the general case, the parental synergy is the weighting factor that determines the impact of the degree of dependency between the parent nodes for a given value assignment, as reflected by $s_{\mathbf{A}^\mathbf{m}} - t_{\mathbf{A}^\mathbf{m}}$ on the size of the convergence error.

6 Discussion

In this paper the notion of parental synergy was introduced. This synergy is computed from the conditional probabilities as specified for a node in a Bayesian network. For a node with binary parents, the parental synergies can be thought of as a measure of the feasible changes in its probability landscape, given a change in the value of one of its parents. Moreover, a general expression for the prior convergence error, was proposed. A prior convergence error arises in the prior marginal probabilities computed for a node when its parents are considered to be independent. This type of error arises in the probabilities computed by the loopy-propagation algorithm; a widely used algorithm for approximate probabilistic inference. The expression of the prior convergence error for a node is composed of the parental synergies of this node and of terms that capture the degree of dependence between its parent nodes. The parental synergy acts as weighting factor determining the impact of the degree of dependency between the parent nodes on the size of the convergence error.

In this paper, the parental synergy just features in the expression of the prior convergence error. Its role as weighting factor in this expression, however, indicates that the parental synergy captures a fundamental characteristic of the probability landscape of a Bayesian network. It is conceivable, therefore, that the parental synergy has a wider range of application.

Acknowledgments

The research reported in this thesis was supported by the Netherlands Organisation for Scientific Research (NWO). I would like to thank Linda van der Gaag for her useful comments on earlier drafts.

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.*

J.H. Bolt. 2008. *Bayesian Networks: Aspects of Approximate Inference.* PhD thesis, Department of Information and Computing Sciences. Utrecht University.

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

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

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

A Score Based Ranking of the Edges for the PC Algorithm

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

Abstract

The result of applying the PC learning algorithm can depend of the order in which independence tests are carried out. Even if these tests are ordered by increasing size of conditional sets, the PC algorithm does not take into account which edges are weaker in order to be considered to be removed before the stronger edges. This paper proposes a new learning algorithm which scores the edges according to a Bayesian metric and adds them to the final graph according to this score. Then, conditional independence tests are carried out to remove edgess as in the PC algorithm. Also, this algorithm is hybridized with a variation of the PC algorithm consisting in determining minimum size cut sets between two nodes to study the deletion of an edge. Some experiments are carried out to evaluate the performance of the new proposals against the PC algorithm.

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). 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 the PC algorithm (Spirtes et al., 1993).

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. 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 a smart selection and ordering of the questions that have to be done in order to recover a causal structure.

In this paper, our basic idea is to combine the PC strategies with additional procedures to improve its performance. In this line, we can cite the work of van Dijk et al. (2003) who propose a combination of order 0 and 1 tests of the PC algorithm with a scoring and searching procedure; Dash and Druzdzel (1999) carry out several PC algorithms with different orders of the variables, which are scored afterwards with a Bayesian metric. In Abellán et al. (2006) we studied several variations of the basic PC strategy. Of them, the best performance was obtained by considering the minimum cut sets to carry out independence tests and changing the Chi-square based tests to Bayesian tests.

In this paper, we propose an algorithm, which considers two graphs: a graph of candidate edges and a graph of added edges (final graph). Each link in the candidates graph is scored with a Bayesian metric. While there are edges with a positive score, the edge with great-

est score is added to the final graph. Each time that an edge is added to the final graph, some conditional independence tests are carried out to deleted links from the candidates graph as in the PC algorithm. These tests are used to update the scores of the candidates graph. We will also test an hybridized version of this algorithm with the algorithm presented in Abellán et al. (2006), consisting in applying the new algorithm but doing only tests of order 0 and 1, and then the final graph is used as the initial structure for the algorithm presented in Abellán et al. (2006) considering minimum cut sets for the independence tests. Tsamardinos et al. (2006) follow a similar idea, but with some differences: the candidate links are locally computed for each single node (instead of following a global procedure) and the procedure finishes with a greedy optimization of a Bayesian score (without the orientation phase of the PC algorithm).

We will describe the new algorithms and we will make some experiments showing their performance when learning Asia and Alarm networks (Beinlich et al., 1989). The quality of the learned networks will be measured by the number of missing-added links and the Kullback-Leibler distance of the learned network to the original one.

The paper is organized as follows: Section 2 is devoted to describe the fundamentals of the PC algorithm and the variations introduced in Abellán et al. (2006). Section 3 introduces the new algorithm and its hybridization; 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 $(\mathbf{A} \perp \mathbf{B}|\mathbf{C})$ we will denote that sets $\mathbf{A}$ and $\mathbf{B}$ are conditionally independent given $\mathbf{C}$.

The 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). The PC algorithm is based on the existence of a procedure which is able to say when $(\mathbf{A} \perp \mathbf{B}|\mathbf{C})$ 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 are 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 $(X \perp Y|\mathbf{S})$
         7. **If** this set exists
            8. Make $S_{XY} = \mathbf{S}$
            9. Remove $X - Y$ edge 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 no link between X and Y, if and only if there is a subset $\mathbf{S}$ of the adjacent nodes of X such that $(X \perp Y|\mathbf{S})$. 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 $(X \perp Y|\mathbf{S})$, 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 are verified. Independencies are decided in the light of 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 $\mathbf{S}$ 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.

In the PC algorithm it is possible that we delete the link between X and Y by testing the independence $(X \perp Y|\mathbf{S})$, when $\mathbf{S}$ 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 (Abellán et al., 2006), we considered 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 consider $CUT_{X,Y} = \{Z\}$. This version of the algorithm will be called BPC algorithm. The computation of this set needs some extra time, but it can be done in polynomial time with a modification of Ford-Fulkerson algorithm (Acid and de Campos, 1996).

The PC algorithm performs a chi-square statistical test to decide about independence.

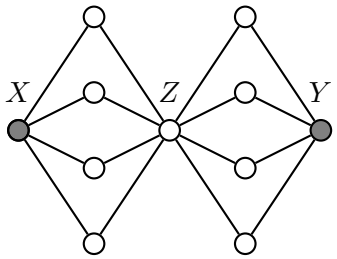


Figure 1: An small cut set

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). Cooper (1997) had proposed a different independence test based on a Bayesian score, but only when conditioning to 0 or 1 variable. In (Abellán et al., 2006) we proposed to do all the statistical tests by using a Bayesian Dirichlet score (Heckerman, 1995) with a global sample size s equal to 1.0. The test $(X \perp Y|\mathbf{S})$ is carried out by comparing the scores of X with $\mathbf{S}$ 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 are considered dependent. The score of X with a set of parents $Pa(X) = \mathbf{Z}$, denoted as $BDe(X, \mathbf{Z})$ 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.

We considered other variations as *refinement* and *triangle resolution*, but they did not prove to be very effective to recover the causal structure.

3 The Score Based PC Algorithm

This algorithm could be carried out with classical statistical tests or with Bayesian scores. In preliminary experiments, the statistical tests did not show a good performance in measuring the strength of an edge, so finally only Bayesian scores have been finally considered.

Instead of starting with a full graph G', now we start with two graphs: a full graph of candidate links G_c and an empty graph of added links G_a. We also have an array T in which we store a numerical value for each link $X-Y$, representing its strength.

The following steps are carried out:

- First, we compute the strength $T[X-Y]$ of each edge, by measuring the Bayesian score of Y conditioned to X minus the score of Y (conditioned to the empty set).
- If the score of an edge is negative, then it is removed from the graph of candidates G_c.
- While there are links in G_c, we select the edge with greatest score $X-Y$. This edge is removed from G_c and added to G_a.
- Once $X-Y$ is added to G_a, we make all the new necessary independence tests for all the candidate edges $Z-V$ in G_c. This is done in the following way:
  - We consider one of the extreme nodes, Z, of the edge $Z-V$ and compute the set of its adjacent nodes in graph G_a such that there is a path from each one of these nodes to the other extreme, V, before adding link $X-Y$ (denoted as $PADJ_Z$) and the set of adjacent nodes to Z such that there is a path from these nodes to V after adding link $X-Y$, but they are not in $PADJ_Z$ (denoted as $NewPADJ_Z$).
  - For each $\mathbf{S} \subseteq PADJ_Z \cup NewPADJ_Z$ such that $\mathbf{S} \cap NewPADJ_Z \neq \emptyset$ we compute the degree of dependence of Z and V given $\mathbf{S}$, by means of the expression: $Dep(Z,V|\mathbf{S}) = BDe(Z,\mathbf{S} \cup \{V\}) - BDe(Z,\mathbf{S})$.
    If this value is less or equal to zero, we can consider that Z is independent of V given $\mathbf{S}$ and the link $Z-V$ is removed from G_c.
    If $Dep(Z,V|\mathbf{S})$ is positive, then $T[Z-V]$ is set to the minimum of its present value and $Dep(Z,V|\mathbf{S})$.
  - If the link is not removed, we repeat above computations for the other extreme V of $Z-V$.

To compute $NewPADJ_Z$, we consider the nodes U in $ADJ_Z - PADJ_Z$, then this node is included in $NewPADJ_Z$ if there is a path from one of the extremes of $X-Y$ to V and a path from U to the other extreme.

The PC algorithm makes the statistical tests in increasing order of the cardinality of the conditional sets, but for the same cardinality the order is arbitrary. It can be the case that we have two links $X-Y$ and $U-V$ and that for a given cardinality, the link that is removed depends of whether X or U is considered first in step 4 of PC algorithm. Here, we try to make less arbitrary this selection, by measuring the strength of the links and adding the strongest links first, leaving the others for candidates to be removed.

Another source of inconsistencies is the following: the PC algorithm starts with a full network and then it removes the links in subsequent steps. When in step 6 of the algorithm, we are testing the independence of X and Y given a subset $\mathbf{S} \subseteq ADJ_X - \{Y\}$, then it is possible that the result is independence and the link $X-Y$ is removed, but later some link connecting X with a node of $\mathbf{S}$ is also removed. So, link $X-Y$ is removed by making a test conditional to a set containing nodes that are not adjacent to X in the final learned graph. This would not be a problem if all the tests were always correct, but the errors are more probable when the size of the conditional sets increases. So, this is a real problem in practice.

The new algorithm has some similarities with K2 greedy algorithm (Cooper and Herskovits, 1992) as adds the edges in increasing order of scoring, but there are some differences: first

it keeps the idea from PC algorithm of carrying out independence tests to remove candidate edges; and it leaves the orientation of the edges for a posterior step (K2 adds oriented links and our algorithm non-oriented edges).

The algorithm does not have the guarantee of recovering a directed acyclic graph under the faithfulness hypothesis and assuming that the statistical tests make always the right decision, as when a link is added to the final graph, it is never considered for deletion again. For having guarantee, it would be necessary that given a graph, the maximum of the scores T computed by the algorithm for a subset of the edges of the candidates graph is always obtained for an edge that is present in the original graph. However, the guarantee can be recovered if after our algorithm, the PC algorithm is applied to graph G_a (instead of starting with a full graph). This is precisely the basis of our hybridized version of the algorithm (called MPC algorithm):

1. To apply our algorithm, but doing only tests of order 1: instead of testing independence for each $\mathbf{S} \subseteq PADJ_Z \cup NewPADJ_Z$ such that $\mathbf{S} \cap NewPADJ_Z \neq \emptyset$, the independence is tested for every set $\mathbf{S} = \{U\}$, where $U \in NewPADJ_Z$.

2. To apply BPC algorithm, starting with graph $G' = G_a$, instead of the full graph, and considering minimum size cut sets.

That is, the idea of links strength is only applied to order 0 and 1 statistical tests. The rest works as in BPC algorithm. The basis is that in the experiments, we can observe that a great numbers of links are deleted precisely with these types of tests. Also, our algorithm had lost the property of doing the statistical tests in increasing cardinality of the conditional sets, and this new hybridized version recovers this property as the first part only carry out tests of order 0 and after order 1 tests. Then we apply BPC algorithm to carry our tests in increasing cardinality (starting with 1, as order 0 tests are not necessary).

4 Experiments

We have done some experiments with the Asia and Alarm networks for testing the PC variations. In all of them, we have generated samples of different sizes by simulation using logic sampling (Henrion, 1988). 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 of the learned probability distribution to the original one[1]. 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 the PC algorithm.

Experiments have been carried out in Elvira environment (Elvira Consortium, 2002). The sample sizes we have used are: 100, 500, 1000, 5000, 10000, and for each sample size, we have repeated the experiment 100 times. The algorithms we have tested are the following:

- The classical PC algorithm with tests done by comparing Bayesian scores.
- The BPC algorithm as introduced in (Abellán et al., 2006) with minimal separating sets and score based tests.
- The new algorithm (HPC) consisting in adding edges in increasing strength value.
- The hybridized version (MPC).

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 for the Asia network, whereas Tables 6, 7, and 8 contain the same data for the Alarm network.

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

	100	500	1000	5000	10000
BPC	3,07	1,75	1,43	0,56	0,45
HPC	2,88	1,78	1,29	0,5	0,36
PC	4,46	3,35	3,23	2,77	2,62
MPC	3,01	1,74	1,34	0,51	0,41

Table 1: Average number of missing links (Asia)

	100	500	1000	5000	10000
BPC	1,61	0,79	0,73	0,19	0,25
HPC	1,4	0,52	0,31	0,15	0,13
PC	0,36	0,17	0,18	0,03	0,06
MPC	1,51	0,87	0,64	0,16	0,18

Table 2: Average number of added links (Asia)

	100	500	1000	5000	10000
BPC	0,2034	0,0842	0,0500	0,0209	0,0248
HPC	0,2147	0,0678	0,0344	0,0175	0,0192
PC	0,2934	0,1883	0,1957	0,2042	0,2020
MPC	0,2248	0,1644	0,1444	0,117	0,076

Table 3: Aver. K-L distance (Asia)

	100	500	1000	5000	10000
BPC	0,0540	0,2548	0,5327	2,9321	6,1866
HPC	0,0414	0,2408	0,5098	2,8342	6,1092
PC	0,0567	0,3508	0,7404	4,3941	9,3542
MPC	0,052	0,2784	0,6056	3,3988	7,1766

Table 4: Average time (Asia)

	100	500	1000	5000	10000
BPC	22,1	11,3333	7,8666	4,5	3,833
HPC	18,3	10,3	7,733	5,4666	5
PC	27,833	18,533	14,566	8,466	7,066
MPC	19,6	9,5	6,1	4,033	3,666

Table 5: Aver. number of missing links (Alarm)

	100	500	1000	5000	10000
BPC	13,9666	7,0333	4,9	3,7666	3,6333
HPC	9,1333	5,6666	4,8666	5,066	5,1333
PC	3,7	0,66	0,3660	0	0,033
MPC	11,2	5,5	2,8666	2,3	3

Table 6: Aver. number of added links (Alarm)

	100	500	1000	5000	10000
BPC	4,4597	2,3188	1,7611	0,8902	0,6839
HPC	3,9895	1,8093	1,3292	0,6411	0,6009
PC	5,0719	3,2705	2,5012	1,1374	0,7460
MPC	4,3594	2,5936	1,8357	0,7917	0,6289

Table 7: Aver. K-L distance (Alarm)

	100	500	1000	5000	10000
BPC	2,8917	8,4428	16,6012	92,8290	197,5764
HPC	0,9107	4,7952	10,5295	62,3101	132,7068
PC	1,1023	6,8248	16,2425	114,1656	252,8078
MPC	1,881	6,6598	13,1989	76,6548	168,1496

Table 8: Average time (Alarm)

In these results we highlight the following facts:

- In the simple Asia network new algorithm (HPC) has a better performance than BPC in terms of added links, deleted links and Kulback-Leibler distance, being more efficient in time, for all the sample sizes. With respect to PC algorithm and the Asia network, HPC has more errors in terms of added links, but the total number of errors (added + missing links) is lower in the new HPC algorithm.

- The hybridized algorithm (MPC) has an intermediate behaviour between BPC and HPC in terms of error and time in the Asia network. The Kullback-Leibler distance is worse than in both algorithms.

- In the Alarm network, the new algorithm is better in terms of total errors (added + missing links) than BPC for small or medium sample sizes (less or equal than 1000). However, the behaviour deteriorates for larger sample sizes. The hybridized algorithm is the best in terms of total errors for medium or larger sample sizes. However, the Kulback-Leibler distance is always lower for the new HPC algorithm. This can be interpreted in the following way: even if for large sample sizes, we can make more errors than in the BPC or MPC algorithms, these errors are less important (for example the missing links represent weaker dependencies).

- The new algorithm is always more efficient in time than all the other algorithms. This may be due to the fact that adding first the more important dependencies and making tests taking them into account, makes these

tests more successful in removing candidate links. The extra time necessary to compute $NewPADJ_Z$ is not as high as the time we save making less independence tests.

We were a bit surprised by the fact that the HPC algorithm does not show the best performance in terms of total number of errors in Alarm network for large sample sizes. Then, we analyzed the type of errors that the algorithm was doing in concrete cases. We have found that the errors are almost the same (same missing or added links) for different databases. In the case of missing links, these are usually one link pointing to one variable with several parents. Furthermore, only for a few combinations of values of the other variables, this new variable is really relevant (for the other combinations, the missing parent has little influence). When we make the test, it should produce dependence and it produces independence in a systematic way (with different databases). Our explanation is the following: assume that we are testing independence of X and Y given $\mathbf{S}$, by comparing the scores $BDe(X, \mathbf{S} \cup \{Y\})$ and $BDe(X, \mathbf{S})$. Both scores, can be expressed as a sum in the different values $\mathbf{s}$ of $\mathbf{S}$. If Y is relevant to X only for some values of $\mathbf{s}$, it is possible that for these values, we have $BDe(X, \mathbf{s} \cup \{Y\}) > BDe(X, \mathbf{s})$, but for the other values we may have the opposite inequality $BDe(X, \mathbf{s} \cup \{Y\}) < BDe(X, \mathbf{s})$. The final score is computed by adding in the different values $\mathbf{s}$. Then, the final result will depend of which of the differences is larger. Then, if we have asymmetrical independencies (or very weak dependencies) for a majority of values of $\mathbf{s}$, then the test will produce an error. This is due to the nature of the test. So, very little can be done by playing with the strategy of ordering the different independence tests.

The case of added links is different. As we have said, even under the faithfulness hypothesis and with no errors in the tests, our HPC algorithm does not have guarantee of recovering the original network, as it is possible that some of the tests necessary for the deletion of a link are never considered. When, this algorithm is combined with the BPC algorithm in the MPC algorithm, then the number of added links decreases, as more additional tests are carried out.

5 Conclusions

In this paper we have proposed two new strategies to organize the statistical tests in the PC algorithm: the HPC and the MPC algorithms. In general, the results of the HPC algorithm are better than in the classical PC algorithm and BPC algorithm proposed in Abellán et al. (2006). Furthermore, the HPC algorithm if the fastest in time. The MPC algorithm is not as good, but it produces the least number of errors in Alarm network for large sample sizes.

We recognize that the experiments in this paper are clearly insufficient and that more extensive experiments are necessary to determine in which conditions is appropriate to apply the new algorithms, but even that we want to highlight two points:

The first one is that though, at the present, general search algorithms are more common in practice, we believe that it is very promising to work in this type of PC based strategies. We believe that there are good opportunities of improving performance in terms of errors and time. In general, algorithms in graphs are fast compared with the time devoted to carry out statistical tests, so it is worthy to make some work in graph computations to save some statistical tests. We believe, that the orientation step could also use some of the ideas of this paper (sometimes there are conflicts among the different rules used for orientation, and it could be useful to decide with the help of an score).

The second point is that some more work is necessary to determine how to carry out the statistical tests. Classical statistical tests have known problems (Moral, 2004; Abellán et al., 2006). But, as we have shown here, Bayesian tests based on scores can produce systematic errors, for example in the case of asymmetrical independencies (or situations in which some of the dependencies are really weak). Perhaps, if for a value $\mathbf{s}$, we have $BDe(X, \mathbf{s} \cup \{Y\}) > BDe(X, \mathbf{s})$,

we should consider X and Y dependent given $\mathbf{S}$, with some correction given that we make multiple comparisons.

Acknowledgments

This work has been jointly supported by the Spanish Ministry of Education and Science under project TIN2007-67418-C03-03, by European Regional Development Fund (FEDER), and by the Spanish research programme Consolider Ingenio 2010: MIPRCV (CSD2007-00018).

References

J. Abellán, M. Gómez-Olmedo, and S. Moral. 2006. Some variations on the PC algorithm. In *Proceedings of the Third European Workshop on Probabilistic Graphical Models (PGM' 06)*, pages 1–8.

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.

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.

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. Gámez and A. Salmerón, 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.

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.

M. Henrion. 1988. Propagating uncertainty by logic sampling in Bayes networks. In J. Lemmer and L.N. Kanal, editors, *Uncertainty in Artificial Intelligence, 2*, pages 149–164. Horth-Holland, Amsterdam.

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.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78.

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

A Bayesian approach to estimate probabilities in classification trees

Andrés Cano and Andrés R. Masegosa and Serafín Moral
Department of Computer Science and Artificial Intelligence
University of Granada
18071, Granada, Spain

Abstract

Classification or decision trees are one of the most effective methods for supervised classification. In this work, we present a Bayesian approach to induce classification trees based on a Bayesian score splitting criterion and a new Bayesian method to estimate the probability of class membership based on Bayesian model averaging over the rules of the previously induced tree. In an experimental evaluation, we show as our approach reaches the performance of Quinlan's C4.5, one of the most known decision tree inducers, in terms of predictive accuracy and clearly outperforms it in terms of better probability class estimates.

1 Introduction

Decision trees or classification trees (decision trees where is predicted a probability of class membership instead of the class label simply) have been one the most used and better studied predictive models. Several reasons appear examining their wide popularity such as their simplicity and their easy interpretability. At the same time, they are fast and effective as classifiers (Lim et al., 2000) even at very large data sets (Provost and Kolluri, 1999) and there have been available several software packages for learning classification trees (CART (Breiman et al., 1984) and C4.5 (Quinlan, 1993)).

One of the main problems of classification trees is the poor estimates of class probabilities they usually produce (Breiman, 1996a; Pazzani et al., 1994). In many situations, a good estimate of the probability class is a strong requirement, specially, when it is needed a ranking of the samples by the class they belong to. E.g., most of the web search engines ranks the web pages based on their probability of being relevant for a given query.

Let us see an example of the problem to assign class probabilities in a classification tree. Suppose we have induced a classification tree for a two-class classification problem and we find that a leaf node defines a subset of 3 samples, all of them belonging to the positive class. Maximum likelihood estimate shall assign a probability of 1.0. The question that arises now if there is enough evidence with 3 samples to assess such a strong statement.

In (Provost and Domingos, 2003) a survey study of different methods for better probability of class estimates was carried out based on C4.5. They compare three different methods: Laplace estimate, C4.5 pruning (Quinlan, 1993) and, specially, bagging (Breiman, 1996b). They conclude with a positive evidence in favor of Laplace and Bagging, but they do not find a definitive conclusion for pruning.

In this work, we present a Bayesian approach to induce classification trees with the aim to maintain the predictive accuracy of one of the state-of-the-art classification tree inducers $J48$ (an advanced version of Quinlan's C4.5) and produce significative improvements in the estimates of probability class beyond the use of Laplace correction or a post-pruning process. We conduct an experimental study over 27 UCI databases to evaluate our Bayesian approach.

The rest of the paper is organized as follows. In Section 2 we introduce the necessary nota-

tions and we briefly explain classification trees and C4.5 tree inducers (Quinlan, 1993). After that, in Section 3 we describe our Bayesian approach to induce classification trees. The experimental evaluation and the results are shown in Section 4. Finally, Section 5 is devoted to the final conclusions and future works.

2 Previous Knowledge

In a classification problem, we have a target or dependent variable C with k cases $(c_1, ..., c_k)$, $|C| = k$, and a set of predictive or independent variables $\mathbf{X} = (X_1, ..., X_n)$. The goal is to induce a probability function for every unseen sample $\mathbf{x}$ to a probability distribution of C: $(\mathcal{L}(\mathbf{X}) \rightarrow \mathcal{P}(C))$. This function is represented as a posterior probability of C given a sample $\mathbf{x}$: $P(C|\mathbf{x}) = P(C|X_1 = x_1, ..., X_n = x_n)$.

This posterior probability has to be inferred from a limited set of samples of the joint distribution $(\mathbf{X}, C)$, the learning data $\mathcal{D}$[1].

2.1 Classification Trees

A classification tree, $\mathcal{T}$ is an efficient representation of this posterior probability of C as a tree recursive structure, such as the one in Figure 1.

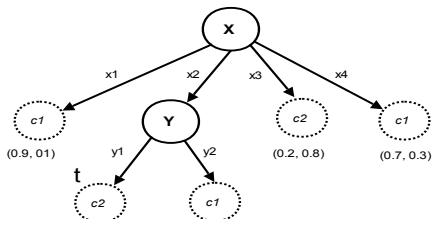


Figure 1: Classification Tree

Each path in the tree from the root node to another descendant node t (not necessarily to a leaf node) defines a configuration $\mathbf{x}^t$ for the variables in $\mathbf{X}^t$, $\mathbf{X}^t \subseteq \mathbf{X}$, where $\mathbf{X}^t$ is the set of variables that labels the internal nodes contained in the path from the root node to the descendant node t without considering the variable at node t. We also denote as $\bar{\mathbf{X}}^t = \mathbf{X} - \mathbf{X}^t$ the set of attributes not included in $\mathbf{X}^t$. E.g., supposing for the tree of Figure 1 that $\mathbf{X} = \{X, Y, Z\}$, we have at the node t: $\mathbf{x}^t = \{X = x_2, Y = y_1\}$, $\mathbf{X}^t = \{X, Y\}$ and $\bar{\mathbf{X}}^t = \{Z\}$.

We also have for each $\mathbf{x}^t$ in the tree $\mathcal{T}$ an estimate of the a posteriori conditioned probability of C given $\mathbf{x}^t$, $\hat{P}_{\mathcal{T}}(C|\mathbf{x}^t)$.

Let us $\mathcal{X}_{\mathcal{T}}$ be the set of configurations $\mathbf{x}^t$ associated to set of nodes (internal or leaf) in a tree $\mathcal{T}$ and $\hat{\mathcal{P}}_{\mathcal{X}_{\mathcal{T}}}$ their associated set of estimated probabilities. We say that a configuration $\mathbf{x}^t \in \mathcal{X}_{\mathcal{T}}$ is *consistent* with a sample $\mathbf{x}$, $\mathbf{x}^t \sim \mathbf{x}$, if for each $X_i \in \mathbf{X}^t$, $\mathbf{x}^t$ and $\mathbf{x}$ contains the same value x_i for X_i. In the previous example, if $\mathbf{x} = \{X = x_2, Y = y_1, Z = z_2\}$ then $\mathbf{x}$ is consistent with $\mathbf{x}^t = \{X = x_2, Y = y_1\}$.

Let us denote as $\mathcal{X}_T^{\mathbf{x}} = \{\mathbf{x}^t \in \mathcal{X}_T : \mathbf{x}^t \sim \mathbf{x}\}$ to the set of configurations $\mathbf{x}^t$ *consistent* with the sample $\mathbf{x}$. It can be seen that this set contains the configurations $\mathbf{x}^t$ of the nodes t in the path from the root node until one of the leaf nodes. This set shall contain p elements, $\mathcal{X}_T^{\mathbf{x}} = \{\mathbf{x}^{t_0}, \mathbf{x}^{t_1}, ..., \mathbf{x}^{t_p}\}$, where t_0 is the root node ($\mathbf{x}^{t_0}$ is an empty configuration), t_1 is the children of t_0 in the previous path ($\mathbf{x}^{t_1}$ is configuration for the variable at the root node) and so on until t_p, which is the leaf node in the previous defined path. For the previous example, $\mathcal{X}_T^{\mathbf{x}} = \{\emptyset, \{X = x_2\}, \{X = x_2, Y = Y_1\}\}$.

The largest configuration, $\mathbf{x}^{t_p}$, is taken to make the classification through the use of the estimated conditional probability $\hat{P}_T(C|\mathbf{x}^{t_p})$.

In order to build or induce a classification tree there are two main elements that need to be defined: a *splitting criterion* and a *stop criterion*. The *splitting criterion* is based on a measure or score, $\mathbf{SC}_t$, to decide which attribute is placed in the node t' of the tree as descendent of the node t. Usually it is taken the attribute $X^t = arg\max_{X_i \in \bar{\mathbf{X}}^t} \mathbf{SC_t}(X_i)$.

The *stop criterion* at the branch of the node t is normally associated with $\mathbf{SC}_t$ and it decides when no more attributes are added to the actual configuration $\mathbf{x}^t$.

We shall use $\mathbf{x}$ instead of $\mathbf{x}^t$ for simplicity reasons when there is no possibility of confusion.

2.2 C4.5: A classification tree inducer

As we have already commented, C4.5 (Quinlan, 1993) is one of the most known and widely used

[1] Absence of missing values and continuous attributes in $\mathcal{D}$ are assumed.

classification tree inducers.

$J48$ is an advanced version of C4.5 implemented in Weka (Witten and Frank, 2000) uses the following score as splitting criterion known as *Info-Gain Ratio* (Quinlan, 1993) based on the *Info-Gain* measure, IG, and the *entropy* measure, H:

$$\mathbf{SC}_t(X) = \frac{IG(X,C)}{H(X|\mathbf{x}^t)}$$

The attributes X that does not verify that its *Info-Gain* measure is greater than the mean value of the *Info-Gain* measure of the rest of candidate variables are discarded. $\mathbf{SC}_t(X) = 0$ if $IG_{\mathbf{x}^t}(X,C) < Average\{IG_{\mathbf{x}^t}(X_i,C) : X_i \in \bar{\mathbf{X}}^t\}$. This C4.5 version stops at $\mathbf{x}^t$ when $\forall X_i \in \bar{\mathbf{X}}^t : SC_{\mathbf{x}^t}(X_i) \leq 0$.

Once the tree is built, C4.5 applies a post-pruning process (Quinlan, 1993) in order to reduce the size of the tree and avoid an overfitting to the data. This pruning process is a single bottom-up pass where the estimated error rate for each subtree is computed as an upper bound of the misclassification rate with a confidence level of 0.25 through the use of a binomial trial process. A subtree is removed if there is not reduction in the estimated error rate. We shall call $C4.5_\rho$ the version with pruning and $C4.5_{\neg\rho}$ the version without pruning.

In both cases, a Laplace estimate is used in order to smooth the probabilities of class membership.

3 A Bayesian Approach for classification tree induction

In this section, our approach is presented. Firstly, we describe the splitting criterion problem as a Bayesian model selection problem. Secondly, we express the problem of probability class estimates as a Bayesian model averaging problem. And finally, we present a method to introduce non-uniform priors in the two previous Bayesian approaches.

3.1 Bayesian Model Selection to Classification Tree induction

Bayesian model selection (BMS) (Wasserman, 2000) is an approach to choose among a set of alternative models in a model space $\mathcal{M} = \{M_1, ..., M_k\}$ where M_i is a set of probability densities with unknown parameters θ_i: $M_i = \{P_{\theta_i}(\mathbf{x}) : \theta_i \in \Omega_i\}$. In this approach the quality of a model is estimated by the posterior probability of the model given the learning data $\mathcal{D}$. Where $P(M|\mathcal{D})$ can be computed through Bayes' rule as:

$$P(M|\mathcal{D}) = \frac{P(\mathcal{D}|M)P(M)}{P(\mathcal{D})} = \frac{P(M)\int \mathcal{L}(\theta)P(\theta)d\theta}{P(\mathcal{D})}$$

where $\mathcal{L}(\theta)$ is the likelihood of the parameters given the data ($P(\mathcal{D}|M,\theta)$) and $P(\theta)$ the prior distributions over the parameters of the model.

The use of Bayesian metrics (Heckerman et al., 1994) are suitable to compute model quality because of the inherent penalty they give to the more complex models in order to prevent against over-fitting and they provide closed-form equations to compute the posterior probabilities of the model given the learning data.

The metric, denoted as $\beta(M)$, computes these probabilities assuming an uniform prior probability over the possible models, $P(M)$, and a prior Dirichlet distribution over the parameters, θ, with independence for the parameters at the different conditional distributions. Usually a global or equivalent sample size, S, is considered and then it is assumed that for each variable X the prior probability about the vector $\sigma_X = (\sigma_1, ..., \sigma_{|X|})_X$ is Dirichlet with the same parameters $\alpha_k = S/|X|$ for all values, where $|X|$ is the number of possible cases of X. $P(\mathcal{D})$ can be also discarded because is the same for all models. So that $P(M|\mathcal{D}) \propto P(\mathcal{D}|M)$. The score will be the value $\beta(M) = P(\mathcal{D}|M)$ (usually the log of this value for computational reasons).

In the classification tree induction process, the model selection problem appears when we have to choose between to stop the branching or to branch by one of the available attributes.

Formally, we are at a given leaf node t. Let us denote by $\mathcal{D}_t$ the learning data $\mathcal{D}$ restricted to $\mathbf{x}^t$ and projected over the variables $(\bar{\mathbf{X}}^t, C)$, where $\bar{\mathbf{X}}^t$ are the variables not included at the node t (or available variables for branching at node t, Section 2.1).

In the model selection problem we face here, our model space problem $\mathcal{M}$ is composed by the following alternatives:

- Model M^t: stop branching at node t.
- For each variable $X \in \bar{\mathbf{X}}^t$, model M_X^t: branch node t by variable X and then stop.

Each one of these models M in $\mathcal{M}$ is scored by $P(\mathcal{D}_t|M)$. In this computation, only the class and available variables $\bar{\mathbf{X}}^t$ are relevant. If we denote by c_i and $\bar{\mathbf{x}}_i^t$ the actual values of class variable and available attributes in case r_i of $\mathcal{D}_t$ and if we assume that each r_i is independent and identically distributed, the scores can be expressed as:

$$P(\mathcal{D}_t|M) = \prod_{r_i \in \mathcal{D}_t} P(c_i, \bar{\mathbf{x}}_i^t|M) = P(\bar{\mathbf{x}}_i^t|M).P(c_i|M, \bar{\mathbf{x}}_i^t)$$

Models M^t and M_X^t only differs in the values $P(c_i|M, \bar{\mathbf{x}}_i^t)$ as they not make any hypothesis about the way in which the rest of the variables are distributed, so that we could assume equally distributed in both cases. Then the score of model M will be proportional to $\prod_{r_i \in \mathcal{D}_t} P(c_i|M, \bar{\mathbf{x}}_i^t)$. In the concrete models we have, we obtain:

- In model M^t, variable C is independent of the rest of variables (no branching). So, $\beta(M^t) = P(\mathcal{D}_t|M^t) \propto \prod_{r_i \in \mathcal{D}_t} P(c_i|M_t)$.
- In models M_X^t, C is only dependent of the branching variable X, so that $\beta(M_X^t) = P(\mathcal{D}_t|M_X^t) \propto \prod_{r_i \in \mathcal{D}_t} P(c_i|M_t, x_i)$, where x_i is the value of the branching variable X in case r_i.

Assuming Dirichlet prior probabilities for the class C with uniform parameters $\alpha_k = S/|C|$, where S is a fixed equivalent sample size, then these values can be obtained with the standard expressions:

$$\beta(M^t) \propto \frac{\Gamma(S)}{\Gamma(S+n^t)} \frac{\prod_k \Gamma(n_{c_k} + \alpha_k)}{\prod_k \Gamma(\alpha_k)}$$

$$\beta(M_X^t) \propto \prod_j^{|X|} \frac{\Gamma(S)}{\Gamma(S+n_{x_j})} \frac{\prod_k \Gamma(n_{c_k,x_j} + \alpha_k)}{\prod_k \Gamma(\alpha_k)}$$

where $\Gamma(.)$ is the gamma function ($\Gamma(\alpha+1) = \alpha \cdot \Gamma(\alpha)$), n_{c_k,x_j} is the number of occurrences of $\{(C = c_k, X = x_j)\}$ in the learning sample at node t, $\mathcal{D}_t$ (analogously for n_{c_k}, n_{x_j}) and n^t is the number of cases in $\mathcal{D}_t$.

In that way, our approach branches at the node leaf t by the variable $X^* = arg\max_{X \in \bar{\mathbf{X}}^t}\{\beta(M_X^t)\}$. It stops branching when $\beta(M^t) > \beta(M_{X^*}^t)$, in other words, when the model without branching has a higher probability.

Another important factor, which will be used when estimating the probabilities at the leaves, is that when variable X^* is selected for branching, as the score is proportional to the posterior probability of the data given the model, we have that the value

$$\mathcal{B}_t = \frac{\beta(M_{X^*}^t)}{\beta(M^t)} = \frac{P(\mathcal{D}_t|M_{X^*}^t)}{P(\mathcal{D}_t|M^t)}$$

As only data in $\mathcal{D}_t$ are relevant at node t, we can assume that the rest of data in $\mathcal{D}$ is equally distributed in this node and that

$$\mathcal{B}_t = \frac{P(\mathcal{D}|M_{X^*}^t)}{P(\mathcal{D}|M^t)} = \frac{P(M_{X^*}^t|\mathcal{D})}{P(M^t|\mathcal{D})}$$

So we also have a value equal to the ratio between the probability of the model branching at t with X^* and not branching at all given the data.

3.2 Bayesian Model Averaging to estimate class probabilities

Most of approaches to predictive learning are based on the assumption of the whole database had been completely generated by one model, the "right" one, ignoring the underlying model decision uncertainty involved in the classifier inducing process. This problem specially happens to model selection processes in classification tree inducers, because as smaller the learning sample size is as more uncertain model selection becomes. In a classification tree, the sample size decreases exponentially with the depth of the branch, so decision of branching at final leaves accumulates a big uncertainty.

Bayesian Model Averaging (BMA) (Wasserman, 2000; Hoeting et al., 1999) provides

a foundation to deal with this uncertainty through the use of a weighting scheme for each candidate hypothesis of the hypothesis space computing its posterior probability given the learning data.

Formally, we start with a hypothesis space $\mathcal{H}$ and a set of learning data $\mathcal{D}$. So, the posterior probability of C given a new sample $\mathbf{x}$ is computed by:

$$P(C|\mathbf{x},\mathcal{D},\mathcal{H}) = \sum_{h\in\mathcal{H}} P(C|\mathbf{x},h)P(h|\mathcal{D})$$

Our application of BMA is an alternative to pruning the final tree: as for each inner nodes t we can compute the value $\mathcal{B}_t$ which is proportional to the ratio of the probability of the model branching by variable X^* and the probability of the model without branching, instead of deciding by one of the models, we can make the average of the probabilities estimated with each one of the models weighted by a value proportional to their probability. This averaging is applied in each leaf node for all the nodes in the path from this node to the root.

One advantage of this application of BMA is that only the final estimation of the probabilities at the leaves change, having only one decision tree structure.

In this way, for each inner node t_i we compute a weight proportional to the posterior probability that the induced tree stops at this point (that is denoted as the hypothesis h^{t_i}) and it is computed using the *Bayes factors* $\mathcal{B}_{t_j}$ in the following way:

$$\hat{P}(h^{t_i}|\mathcal{D}) \propto \prod_{j=1}^{i-1} \mathcal{B}_{t_j} = \prod_{j=1}^{i-1} \frac{P(M^{t_j}_{X^*}|\mathcal{D})}{P(M^{t_j}|\mathcal{D})}$$

where $\hat{P}(h^{t_1}|\mathcal{D}) = 1$.

So, the estimated probability in a leaf node t_p is computed as follows:

$$P(c_k|\mathbf{x}^{t_p}) \propto \sum_{i=1}^{p} \frac{n_{c_k\mathbf{x}^{t_i}} + \alpha_k}{n_{\mathbf{x}^{t_i}} + S} \hat{P}(h^{t_i}|\mathcal{D})$$

where $n_{\mathbf{x}^{t_i}}$ is the size of the learning sample at the node t_i, $\mathcal{D}_{t_i}$, and $n_{c_k\mathbf{x}^{t_i}}$ is the number of occurrences of $\{C = c_k\}$ in this set $\mathcal{D}_{t_i}$. The α_k and S values correspond to the same Dirichlet prior probability used in the induction process of the previous Section 3.1. Finally, a normalization is required.

This approach has the advantage that all these probabilities can be efficiently computed at the same time that the tree is built, with only a linear increasing in the complexity.

3.3 A non-Uniform Priors Approach

In all the previous developments we have assumed uniform values, $\alpha_k = S/|C|$, for the parameters of the prior Dirichlet distributions. In this subsection, we are giving a new approach to define non-uniform priors in the induction of decision trees, which will be incorporated in the computation of the Bayesian split criterion, Section 3.1, and in the computation of averaging probabilities, Section 3.2, trough the definition of new α_k values.

To justify this approach, we start with the following idea: if at some node t_i the frequency n_{c_k} is zero, then $\forall j > i$ at t_j descendant nodes the frequency n_{c_k} shall also be zero. So it makes sense to assume that $n_{c_k\mathbf{x}}$ will probably be zero or close to zero for most of future samples $\mathbf{x}$. So decreasing the prior probability for c_k at $\mathbf{x}^{t_{i+1}}$ is coherent.

We propose the following heuristic to modify the parameters of the Dirichlet priors distributions: Let us $\delta_i = |\{n_{c_k} = 0 : c_k \in C\}|$ in $\mathcal{D}_{t_i}$, if $\delta_i > 1$ we define $\alpha_k^{t_{i+1}}$ as follows:

$$\alpha_k^{t_{i+1}} = \frac{S}{(|C| - \delta_i + 1)} : n_{c_k\mathbf{x}^{t_i}} \neq 0$$

$$\alpha_k^{t_{i+1}} = \frac{S}{(|C| - \delta_i + 1)\delta_i} : n_{c_k\mathbf{x}^{t_i}} = 0$$

As we can see, those cases with non-null frequency have the same prior probability, $\frac{1}{(|C|-\delta_i+1)}$, while all those cases with null frequency share among them the same probability mass of one non-null frequency case, i.e., $\frac{1}{(|C|-\delta_i+1)\delta_i}$. Let us point out that for a two-class problem we get with this heuristic a uniform prior.

4 Experimental Results

In this section, we present the experimental evaluation of our approach. Firstly, the evaluation methodology is described and, after that, the experimental results of the different approaches are presented.

4.1 Evaluation Methodology

We used the following 27 databases from the UCI repository: *anneal, audiology, autos, breast-cancer, colic, credit-german, diabetes-pima, glass-2, hepatitis, hypothyroid, ionosphere, kr-vs-kp, labor, letter, lymph, mfeat-pixel, mushrooms, optdigits, segment, sick, solar-flare, sonar, soybean, sponge, vote, vowel and zoo*. The features of the databases are very different among them: from 2 to 24 class cases, from 57 to 20000 samples and from 9 to 240 attributes.

The classification tree inducers were implemented in Elvira platform (Consortium, 2002) and evaluated in Weka (Witten and Frank, 2000). And then we preprocessed the data replacing the missing values (with the mode value for nominal attributes and with the mean value for continuous attributes) and discretized with an equal-frequency method with 5 bins using Weka's own filters.

Two evaluation or performance measures are employed in this experimental evaluation: the classical prediction accuracy (noted as *Accuracy*); and the logarithm of the likelihood of the true class, computed as: *log-likelihood*= $\ln(\hat{P}(c_{r_i}|\mathbf{x}))$, where c_{r_i} is the true class value of the example of the test data set. This last score is introduced with the aim to evaluate the precision of probability class estimates. The usefulness of this score for this task is justified in many ways, as for example in (Roulston and Smith, 2002; Gneiting and Raftery, 2005).

The evaluation of the classifiers was achieved by a 10-fold-cross validation repeated 10 times scheme for each database. So, 100 train and test evaluations are carried out. With these estimates, the comparison among classifiers was achieved using a corrected paired t-test (Nadeau and Bengio, 2003) implemented in Weka with a 1% of statistically significant level. In this way, a classifier is fixed as reference (marked with $\star$) and then each proposed classifier is compared against it. The comparison is made summing up the times that the proposed classifier gets a statistically significant difference respect to the reference classifier in accordance with the corrected paired t-test in a given database. The test result can show an statistically significant improvement or Win (marked with W), a not statistically significant difference or Tie (marked with T) and a statistically significant deterioration (marked with D) in the evaluation measures. These results are shown in the rows starting with *W/T/D*. E.g., in Table 1, $\beta_{S=1}$ gets a statistically improvement or win in the accuracy respect to $C4.5_\rho$ in 3 databases, there is no differences in 23 databases and it looses or gets a significant deterioration in the accuracy respect to $C4.5_\rho$ in 1 databases.

As it is commented, our approach is three fold: a Bayesian model selection (BMS) approach as splitting criterion, Section 3.1; a Bayesian model averaging approach (BMA) to estimate the probabilities class membership, Section 3.2; and a non-uniform prior (NUP) definition approach, Section 3.3. In all cases, we use the same prior Dirichlet distributions with the same global sample size, S.

Let us define the four combinations we evaluate: β_S: only BMS; $\widehat{\beta}_S$: BMS + BMA; β_S^θ: BMS + NUP; $\widehat{\beta}_S^\theta$: BMS + BMA + NUP.

In all cases, three different global sample sizes are evaluated: $S = 1$, $S = 2$ and $S = |C|$.

4.2 Bayesian Metric as splitting criterion for inducing CT

We test the use of a Bayesian metric as a splitting criterion for inducing classification trees (CT), previously describe in Section 3.1. In order to compare its efficiency as a CT inducer, we compare, in Table 1, the performance of their induced trees respect to the trees induced by $C4.5$ with pruning ($C4.5_\rho$) and without pruning process ($C4.5_{\neg\rho}$).

The results of Table 1 show as the use of a Bayesian metric with $S = 1$ and $S = 2$ as splitting criterion is competitive to $C4.5_\rho$ and $C4.5_{\neg\rho}$ in terms of accuracy: it wins in three to five databases (always databases with high number

Table 1: Bayesian metric as Splitting Criterion

Classifier	$\star C4.5_{\rho}$	$\beta_{S=1}$	$\beta_{S=2}$	$\beta_{S=\|C\|}$
Accuracy	85.50	85.30	85.56	84.03
W/T/D		3/23/1	3/24/0	1/23/3
log-likelih.	-0.79	-0.79	-0.78	-0.78
W/T/D		4/20/3	6/20/1	5/21/1

Classifier	$\star C4.5_{\neg\rho}$	$\beta_{S=1}$	$\beta_{S=2}$	$\beta_{S=\|C\|}$
Accuracy	84.08	5/22/0	5/22/0	1/24/2
log-likelih.	-0.84	7/19/1	9/17/1	9/17/1
Tree Size	482.5	482.1	459.9	319.6

of classes) and it looses once in *sick* database (a very imbalanced two class data set). In terms of log-likelihood the behavior of β_S is competitive and slightly better (significance differences in 6 databases of $S = 2$), which is important considering the absence of a complex and costly pruning process of our approach.

It is interesting to see as the tree size is quite similar to $C4.5_{\neg\rho}$ for $S = 1$ and $S = 2$. But, obviously, it is greater than $C4.5_{\rho}$ average tree size: 306.6 nodes.

4.3 Bayesian Model Averaging for probability class estimate

Here it is evaluated the introduction of our BMA approach, $\widehat{\beta}_S$. Firstly, we compare this approach versus its respective version without probability averaging, β_S, using the same S in each case. It is also evaluated respect to $C4.5_{\rho}$.

Table 2: Bayesian Model Averaging

Classifier	$\star\beta_S$	$\widehat{\beta}_{S=1}$	$\widehat{\beta}_{S=2}$	$\widehat{\beta}_{S=\|C\|}$
Accuracy		85.50	85.82	83.98
W/T/D		0/27/0	0/27/0	0/23/1
log-likelih.		-0.63	-0.63	-0.77
W/T/D		12/14/1	14/11/2	4/20/3

Classifier	$\star C4.5_{\rho}$	$\widehat{\beta}_{S=1}$	$\widehat{\beta}_{S=2}$	$\widehat{\beta}_{S=\|C\|}$
Accuracy		3/24/0	3/24/0	1/23/3
log-likelih.		12/15/0	11/15/1	4/22/1

Results are presented in Table 2. As we can see, the introduction of the BMA approach do not produce an improvement in terms of accuracy (although it avoids the defeat versus $C4.5_{\rho}$ for $S = 1$). But, in terms of log-likelihood, there is a clear outperforming for $S = 1$ and $S = 2$ respect to the basic version, β_S, and respect to $C4.5_{\rho}$, excepting $S = |C|$.

4.4 Non-Uniform Dirichlet Priors Definition

Finally, we test the introduction of non-uniform priors in both Bayesian approaches for model selection and model averaging. In Table 3 the comparative results are divided in 4 folds.

Firstly, we evaluate the introduction of non-uniform priors in the Bayesian metric as splitting criterion, β_S^{θ}, where it is compared respect to the basic version, β_S, with the same S value. In the second part, it is evaluated the non-uniform priors definition at the BMA approach, $\widehat{\beta}_S^{\theta}$, also comparing against the basic version, $\widehat{\beta}_S$. And in the last two parts, we compare the full approach $\widehat{\beta}_S^{\theta}$ respect to the two versions of $C4.5$.

Table 3: Non-Uniform Priors Definition

Classifier	$\star\beta_S$	$\beta_{S=1}^{\theta}$	$\beta_{S=2}^{\theta}$	$\beta_{S=\|C\|}^{\theta}$
Accuracy		85.64	85.82	85.20
W/T/D		2/25/0	2/25/0	3/24/0
log-likelih.		-0.82	-0.81	-0.78
W/T/D		0/21/6	0/21/6	0/25/2

Classifier	$\star\widehat{\beta}_S$	$\widehat{\beta}_{S=1}^{\theta}$	$\widehat{\beta}_{S=2}^{\theta}$	$\widehat{\beta}_{S=\|C\|}^{\theta}$
Accuracy		85.85	86.04	85.37
W/T/D		2/25/0	2/25/0	4/23/0
log-likelih.		-0.61	-0.60	-0.69
W/T/D		3/23/0	5/21/0	10/17/0

Classifier	$\star C4.5_{\rho}$	$\widehat{\beta}_{S=1}^{\theta}$	$\widehat{\beta}_{S=2}^{\theta}$	$\widehat{\beta}_{S=\|C\|}^{\theta}$
Accuracy		3/24/0	4/23/0	0/27/0
log-likelih.		13/14/0	11/15/1	10/16/1

Classifier	$\star C4.5_{\neg\rho}$	$\widehat{\beta}_{S=1}^{\theta}$	$\widehat{\beta}_{S=2}^{\theta}$	$\widehat{\beta}_{S=\|C\|}^{\theta}$
Accuracy		7/20/0	6/21/0	3/24/0
log-likelih.		12/15/0	12/14/1	11/15/1
Tree Size		596.2	591.2	491.2

Summarizing, we find that the effect of the introduction of non-uniform priors is more clear at $\widehat{\beta}_S^{\theta}$ comparison. It supposes a slight improvement in terms of accuracy (2 wins) but strong in terms of log-likelihood (specially for $S = |C|$). The evaluation respect to $C4.5_{\rho}$ remains quite

similar for $S = 1$ and $S = 2$ but suppose an important enhancement for $S = |C|$.

A close review of the results at database level indicates that the introduction of non-uniform priors implies better estimates in those databases where the Bayesian approach with uniform priors has already achieved them. That is to say, non-uniform priors is suitable for databases with a high number of classes.

5 Conclusions and Future Works

We have presented a new method to induce classification trees with a Bayesian model selection approach as split criterion and with a Bayesian model averaging approach to estimates probability class. We also introduce a new approach to define non-uniform priors over the parameters of the models.

We have carried out an experimental evaluation over 27 different UCI data sets comparing against one of the state-of-the-art tree inducers, $J48$. We have shown as these approaches suppose an slight but robust improvement in terms of accuracy, while all of them offer an important improvement in terms of better probability class estimates by the induced decision trees.

Acknowledgments

This work has been jointly supported by the Spanish Ministry of Education and Science under project TIN2007-67418-C03-03, by European Regional Development Fund (FEDER), the FPU scholarship programme (AP2004-4678) and by the Spanish Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía under project TIC-276.

References

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and Regression Trees*. Wadsworth International Group, Belmont.

L. Breiman. 1996a. Out-of-bag estimation. *Private communication*.

Leo Breiman. 1996b. Bagging predictors. *Mach. Learn.*, 24(2):123–140.

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

Tilmann Gneiting and Adrian E. Raftery. 2005. Strictly proper scoring rules, prediction, and estimation. *Technical Report. Department of Statistics, University of Washington*, 463R.

David Heckerman, Dan Geiger, and David Maxwell Chickering. 1994. Learning bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96.

Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. 1999. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417.

Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. 2000. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach. Learn.*, 40(3):203–228.

Claude Nadeau and Yoshua Bengio. 2003. Inference for the generalization error. *Mach. Learn.*, 52(3):239–281.

Michael J. Pazzani, Christopher J. Merz, Patrick M. Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. 1994. Reducing misclassification costs. In *ICML*, pages 217–225.

Foster Provost and Pedro Domingos. 2003. Tree induction for probability-based ranking. *Mach. Learn.*, 52(3):199–215.

Foster Provost and Venkateswarlu Kolluri. 1999. A survey of methods for scaling up inductive algorithms. *Data Min. Knowl. Discov.*, 3(2):131–169.

J.R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco.

M. S. Roulston and L.A. Smith. 2002. Evaluating probabilistic forecasts using information theory. *Monthly Weather Review*, 130:1653–1660.

Larry Wasserman. 2000. Bayesian model selection and model averaging. *J. Math. Psychol.*, 44(1):92–107.

Ian H. Witten and Eibe Frank. 2000. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Efficient Model Evaluation in the Search-Based Approach to Latent Structure Discovery

Tao Chen, Nevin L. Zhang and Yi Wang
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Kowloon, Hong Kong, China

Abstract

Latent tree (LT) models are tree-structured Bayesian networks where leaf nodes are observed while internal nodes are hidden. We are interested in learning LT models through systematic search. A key problem here is how to efficiently evaluate candidate models during search. The problem is difficult because there is a large number of candidate models, the candidate models contain latent variables, and some of those latent variables are foreign to the current model. A variety of ideas for attacking the problem have emerged from the literature. In this paper we observe that the ideas can be grouped into two distinct approaches. The first is based on data completion, while the second is based on what we call maximum restricted likelihood. We investigate and compare the two approaches in the framework of EAST, a newly developed search procedure for learning LT models.

1 Introduction

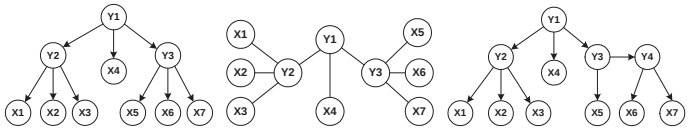
Learning Bayesian networks (BNs) from data has been the focus of much research over the past two decades. Deep insights have been gained for the case where all variables are observed (e.g. Chickering 2002). However, relatively little progress has been made for the case with latent variables. This is not due to the lack of interest on the problem, but its difficulty.

Imagine a search-based algorithm for learning BNs with latent variables. At each step, the algorithm would evaluate a potentially large number of candidate models. Assume the BIC score is used for model selection.[1] To calculate the BIC score of a candidate model m', one needs to compute its maximum loglikelihood $\max_{\theta'} \log P(\mathcal{D}|m', \theta')$, where $\mathcal{D}$ is the data set and θ' is the parameter vector for m'. This requires the expectation-maximization (EM) algorithm. The difficulty is that running EM on a large number of candidate models is computationally prohibitive.

[1]The BICe score suggested by Geiger *et al.* (1996) is currently impractical due to the lack of efficient methods for computing effective model dimension.

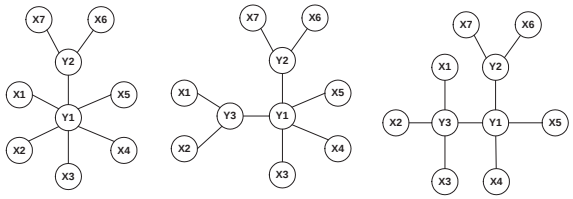
Two methods for overcoming the difficulty have emerged from the literature. The first method is based on the idea of data completion. It first completes the data set $\mathcal{D}$ based on the current model m and uses the completed data set $\bar{\mathcal{D}}$ to evaluate the candidate models. When all the variables in a candidate model m' are observed with respect to $\bar{\mathcal{D}}$, one can evaluate the model using the maximum expected loglikelihood $\max_{\theta'} \log P(\bar{\mathcal{D}}|m', \theta')$ (Friedman 1997). When a candidate model contains variables that are not observed with respect to $\bar{\mathcal{D}}$, one can evaluate the model using heuristics that are computed from $\bar{\mathcal{D}}$ (Zhang and Kočka 2004, Elidan and Friedman 2005).

The second method is based on what we call maximum restricted likelihood. A candidate model m' typically shares many parameters with the current model m. Suppose we have computed the maximum likelihood estimation (MLE) of the parameters of m. Let θ_1^* be the MLE for the parameters that m has in common with m'. Let δ_2 be the parameters of m' that are not shared with m. The method approximates $\max_{\theta'} \log P(\mathcal{D}|m', \theta')$ us-



(a) m (rooted) (b) m (unrooted) (c) m'

Figure 1: Rooted and unrooted latent tree models.



m_1 m_2 m_3

Figure 2: The NI and NR operators.

ing the *maximum restricted loglikelihood*, i.e. $\max_{\delta_2} \log P(\mathcal{D}|m', \theta_1^*, \delta_2)$, and uses the latter to evaluate m'. This method is implicitly used in (Zhang and Kočka 2004). Similar ideas are used in, among others, phylogenetic tree reconstruction (Guindon and Gascuel 2003) and learning of continuous Bayesian networks (Nachman *et al.* 2004).

In this paper we investigate and compare the two methods in the context of latent tree models. Called *hierarchical latent class models* previously (Zhang 2004), *latent tree (LT) models* are tree-structured Bayesian networks where variables at leaf nodes are observed and are hence called *manifest variables*, while variables at internal nodes are hidden and hence are called *latent variables*. All variables are assumed discrete. Figure 1 (a) shows the structure of an LT model.

LT models are interesting for three reasons. First, they relax the local independence assumption of latent class (LC) models (Lazarsfeld and Henry 1968) and hence offer a more general framework for cluster analysis of categorical data (Zhang 2004). Second, LT analysis can reveal latent structures behind data. In this sense LT analysis is a generalization of phylogenetic tree reconstruction (Durbin *et al.* 1998). Using LT models, researchers have found interesting latent structures from stocks data (Elidan and Friedman 2005), marketing data (Zhang 2007) and medical data (Zhang *et al.* 2008). Third, LT models are computationally simple to handle and at the same time can model complex interactions among manifest variables (Pearl 1988). Those two properties make LT models a good tool for density estimation for discrete variables.

2 Search for Optimal Model

Suppose there is a data set $\mathcal{D}$ on a set of manifest variables. To *learn an LT model* from $\mathcal{D}$ means to find a model m that maximizes the BIC score:

$$BIC(m|\mathcal{D}) = \max_{\theta} \log P(\mathcal{D}|m, \theta) - \frac{d(m)}{2} \log N,$$

where θ denotes the set of model parameters, $d(m)$ the number of independent parameters, and N the sample size. It has been shown that edge orientations cannot be determined from data (Zhang 2004). So we can learn only *unrooted latent tree models*, which are latent tree models with all directions on the edges dropped. An example is given in Figure 1 (b). From now on when we speak of latent tree models we always mean unrooted latent tree models unless it is explicitly stated otherwise.

In this section we present a search procedure for learning LT models. Called EAST, the procedure uses five search operators, namely *state introduction (SI)*, *node introduction (NI)*, *node relocation (NR)*, *state deletion (SD)*, and *node deletion (ND)*. They are borrowed from (Zhang and Kočka 2004) with minor modifications. Given an LT model and a latent variable in the model, the *SI* operator creates a new model by adding a new state to the domain of the variable. The *SD* operator does the opposite. The *NI* operator involves one latent node Y and two of its neighbors. It creates a new model by introducing a new latent node Y' to mediate the latent variable and the two neighbors. The cardinality of Y' is set to be that of Y. In m_1 of Figure 2, introducing a new node Y_3 to mediate Y_1 and its neighbors X_1 and X_2 results in m_2. For the sake of computational efficiency, we do not consider introducing a new node to mediate Y and more than two of its

neighbors. The *ND* operator is the opposite of NI. The *NR* operator involves two latent nodes Y_1 and Y_2 and a neighbor Z of Y_1. It creates a new model by relocating Z to Y_2, i.e. removing the link between Z and Y_1 and adding a link between Z and Y_2. In m_2 of Figure 2, relocating X_3 from Y_1 to Y_3 results in m_3.

The search operators can be divide into three groups. NI and SI make the current model more complex and hence are *expansion operators*. ND and SD make the current model simpler and hence are *simplification operators*. NR rearranges connections between the variables and hence is an *adjustment operator*.

The BIC score consists of two terms. The first term measures model fit while the second term penalizes model complexity. Our objective is to optimize the BIC score. Suppose we start with a model that does not fit the data at all. Then improving model fit is the first priority at the initial stage of search. So we first improve model fit by searching with the expansion operators. When the BIC ceases to increase, we switch to the simplification operators in order to minimize the penalty term. The process repeats itself until model score ceases to increase. This is similar to the idea behind the greedy equivalence search (GES) algorithm (Chickering 2002) for learning Bayesian networks.

In the following, we introduce several modifications to the above simple scheme and eventually obtain the EAST procedure given in Figure 3. To understand the first modification, consider the three models in Figure 2. Due to the constraint imposed on NI, it is impossible to reach m_3 directly from m_1. A natural remedy is to consider node relocations after each application of NI. Suppose we have just applied NI to m_1 and have obtained m_2. What we do next is to consider repeatedly relocating the other neighbors of Y_1 in m_1, i.e. X_3, X_4, X_5 and Y_2, to the new latent variable Y_3. This is the `localAdjust`$(m_2, m_1, \mathcal{D})$ subroutine.

The second modification is about choosing between candidate models generated by NI and SI. Let m be the current model and m' be a candidate model. Define the *improvement ratio of m' over m* to be

$$IR(m', m|\mathcal{D}) = \frac{BIC(m'|\mathcal{D}) - BIC(m|\mathcal{D})}{d(m') - d(m)}.$$

It is the increase in model score per unit increase in model complexity. The *cost-effectiveness* principle (Zhang and Kočka 2004) states that, among all candidate models generated by SI and NI, choose the one that has the highest improvement ratio.

We have now completed explaining the `expand` subroutine in Figure 3. The other subroutines are more or less straightforward. The `simplify` subroutine first repeatedly applies SD to the current model until the BIC score ceases to increase and then it does the same with ND. The `adjust` subroutine repeatedly applies the NR operator to the current model until it is no longer beneficial to do so. Unlike in `localAdjust`, there is no restriction on the NR operator here. One can relocate a node to anywhere in the current model. This is an effective mechanism to avoid local maxima.

Located at the top of Figure 3 is the EAST procedure itself. The name EAST is a shorthand for *Expansion, Adjustment, Simplification until Termination*. The procedure takes a data set and an initial model as inputs. It first runs the `expand` subroutine on the initial model. Then it `adjust`s connections between nodes. Thereafter, it passes the resultant model to the `simplify` subroutine. If model score is improved in any of the three steps, the entire process repeats itself with the best model obtained as the initial model.

EAST is similar to the search procedure described in (Zhang and Kočka 2004). There are two main differences. The latter groups NR with NI and SI and hence does not have a separate model adjustment phase. It also restricts how far one can relocate a node.

3 Model Evaluation based on Restricted Likelihood

Each of the arg max operators in EAST evaluates a potentially large number of candidate models. In this section we describe the restricted likelihood method for doing this efficiently.

`EAST`$(m, \mathcal{D})$
Repeat until termination:
  $m_1 \leftarrow$ `expand`$(m, \mathcal{D})$.
  $m_2 \leftarrow$ `adjust`$(m_1, \mathcal{D})$.
  $m_3 \leftarrow$ `simplify`$(m_2, \mathcal{D})$.
  If $BIC(m_3|\mathcal{D}) \leq BIC(m|\mathcal{D})$, return m;
  Else $m \leftarrow m_3$.

`expand`$(m, \mathcal{D})$
Repeat until termination:
  $m_1 \leftarrow \arg\max_{m' \in NI(m) \cup SI(m)} IR(m', m|\mathcal{D})$.
  If $BIC(m_1|\mathcal{D}) \leq BIC(m|\mathcal{D})$, return m.
  If $m_1 \in SI(m)$, $m \leftarrow m_1$;
  Else $m \leftarrow$ `localAdjust`$(m_1, m, \mathcal{D})$

`adjust`$(m, \mathcal{D})$
Repeat until termination:
  $m_1 \leftarrow \arg\max_{m' \in NR(m)} BIC(m'|\mathcal{D})$.
  If $BIC(m_1|\mathcal{D}) \leq BIC(m|\mathcal{D})$, return m;
  Else $m \leftarrow m_1$.

`simplify`$(m, \mathcal{D})$
Repeat until termination:
  $m_1 \leftarrow \arg\max_{m' \in SD(m)} BIC(m'|\mathcal{D})$.
  If $BIC(m_1|\mathcal{D}) \leq BIC(m|\mathcal{D})$, break;
  Else $m \leftarrow m_1$.
Repeat until termination:
  $m_1 \leftarrow \arg\max_{m' \in ND(m)} BIC(m'|\mathcal{D})$.
  If $BIC(m_1|\mathcal{D}) \leq BIC(m|\mathcal{D})$, return m;
  Else $m \leftarrow m_1$.

Figure 3: The EAST search procedure.

Conceptually EAST works with unrooted LT models. In implementation, however, we represent unrooted models as rooted models. Rooted LT models are BNs and their parameters are clearly defined. This makes it easy to see how the parameter composition of a candidate model m' is related to that of the current model m. Consider the models m and m' in Figure 1 (a) and (c). m' is obtained from m by introducing a new latent variable Y_4 to mediate Y_3 and two of its neighbors X_6 and X_7. The two models share the parameters for describing the distributions $P(Y_1)$, $P(Y_2|Y_1)$, $P(X_1|Y_2)$, $P(X_2|Y_2)$, $P(X_3|Y_2)$, $P(X_4|Y_1)$, $P(Y_3|Y_1)$ and $P(X_5|Y_3)$. On the other hand, the parameters for describing $P(Y_4|Y_3)$, $P(X_6|Y_4)$ and $P(X_7|Y_4)$ are peculiar to m' while those for describing $P(X_6|Y_3)$ and $P(X_7|Y_3)$ are peculiar to m.

We write the parameters of a candidate model m' as a pair (δ_1, δ_2), where δ_1 is the collection of parameters that m' shares with m. Similarly, we write the parameters of the current model m as a pair (θ_1, θ_2), where θ_1 is the collection of parameters that m shares with m'. Suppose we have computed MLE (θ_1^*, θ_2^*) of the parameters of m. For a given value of δ_2, $(m', \theta_1^*, \delta_2)$ is a fully specified BN. In this BN, we can compute

$$P(\mathcal{D}|m', \theta_1^*, \delta_2) = \prod_{\mathbf{d} \in \mathcal{D}} P(\mathbf{d}|m', \theta_1^*, \delta_2).$$

As a function of δ_2, this will be referred to as the *restricted likelihood function* of δ_2. The *maximum restricted loglikelihood*, or simply the *maximum RL*, of the candidate model m' is defined to be

$$\max_{\delta_2} \log P(\mathcal{D}|m', \theta_1^*, \delta_2).$$

The *restricted likelihood (RL)* method for model evaluation replaces the likelihood term in the BIC score of m' with its maximum RL and uses the resulting approximate score to evaluate m'.

Local EM is a procedure for computing the maximum RL of a candidate model m'. It works in the same way as EM except with the value of δ_1 fixed at θ_1^*. It starts with an initial value $\delta_2^{(0)}$ for δ_2 and iterates. After $t-1$ iterations, it obtains $\delta_2^{(t-1)}$. At iteration t, it completes the data $\mathcal{D}$ using the BN $(m', \theta_1^*, \delta_2^{(t-1)})$, calculates some sufficient statistics, and therefrom obtains $\delta_2^{(t)}$. Suppose the parameters δ_2 of m' describe distributions $P(Z_j|W_j)$ $(j = 1, \ldots, \rho)$. The distributions $P(Z_j|W_j, \delta_2^{(t)})$ that make up $\delta_2^{(t)}$ can be obtained in two steps:

- E-Step: For each data case $\mathbf{d} \in \mathcal{D}$, make inference in the BN $(m', \theta_1^*, \delta_2^{(t-1)})$ to compute $P(Z_j, W_j|\mathbf{d}, m', \theta_1^*, \delta_2^{(t-1)})$.
- M-Step: Obtain
  $P(Z_j|W_j, \delta_2^{(t)}) = f(Z_j, W_j)/\sum_{Z_j} f(Z_j, W_j)$,
  where the *sufficient statistic*
  $f(Z_j, W_j) = \sum_{\mathbf{d} \in \mathcal{D}} P(Z_j, W_j|\mathbf{d}, m', \theta_1^*, \delta_2^{(t-1)})$.

Local EM converges. The series of loglikelihood $\log P(\mathcal{D}|m', \theta_1^*, \delta_2^{(t)})$ increase monotonically with t. This fact can be seen if one examines the proof for the same result about EM. Like EM, local EM might converge to local maxima.

When implemented properly, local EM is much more efficient than EM. There are two reasons. First, fewer sufficient statistics are computed in local EM than in EM. EM requires sufficient statistics for each pair of neighboring variables, while local EM needs the sufficient statistics only for those pairs that are affected by the search operation. Second,

there are abundant opportunities for computation sharing. Consider calculating the posterior $P(Z_j, W_j|\mathbf{d}, m', \theta_1^*, \delta_2^{(t-1)})$ in a candidate model $(m', \theta_1^*, \delta_2^{(t-1)})$. The candidate model is the same as the current model $(m, \theta_1^*, \theta_2^*)$ except for the part affected by the search operation. Consequently, some of the computational steps can be pre-computed in the current model, and they can be shared at all iterations and among different candidate models.

Local maxima is an issue in local EM as well as in EM. To avoid local maxima, we adopt the *pyramid scheme* proposed by Chickering and Heckerman (1997). The idea is to randomly generate μ initial values for the new parameters δ_2, resulting in μ initial models. One local EM iteration is run on all the models and afterwards the bottom $\mu/2$ models are discarded. Then two local EM iterations are run on the remaining models and afterwards the bottom $\mu/4$ models were discarded. Then four local EM iterations are run on the remaining models and so on. The process continues until there is only one model. After that some more local EM iterations are run on the only remaining model to refine the parameters until the total number of iterations reaches a predetermined number ν.

We have described the RL method for implementing the arg max operators in EAST. It takes a list of candidate models as input and outputs a single model. Full EM is run on the output model to optimize its parameters before the search moves on to the next step.

4 Model Evaluation based on Data Completion

We now turn to the *data completion (DC)* method for efficient model evaluation. The method first completes the data set $\mathcal{D}$ using the current model (m, θ^*), where θ^* is the MLE of the parameters of m. It then uses the completed data set $\bar{\mathcal{D}}$ to evaluate candidate models. The completed data set is used in different ways in different subroutines of EAST.

A candidate model m' in the `adjust` subroutine shares the same variables as m. All the variables are observed with respect to $\bar{\mathcal{D}}$. Hence it is easy to compute the maximum expected loglikelihood $\max_{\theta'} \log P(\bar{\mathcal{D}}|m', \theta')$. The DC method replaces the first term of the BIC score by the maximum expected loglikelihood and uses the resulting function to evaluate m'. This idea is due to Friedman (1997) and it is also used in the `localAdjust` subroutine.

Next consider a candidate model m' in the second loop in the `simplify` subroutine. Suppose it is obtained from m by deleting a latent node Z. Let $\bar{\mathcal{D}}_Z$ be the data set obtained from $\bar{\mathcal{D}}$ by removing the values for Z. Then all the variables in m' are observed with respect to $\bar{\mathcal{D}}_Z$. The DC method replaces the first term of the BIC score by $\max_{\theta'} \log P(\bar{\mathcal{D}}_Z|m', \theta')$ and uses the resulting function to evaluate m'. This idea is from Zhang and Kočka (2004).

The SD operator deletes a state from the domain of a latent variable. A related operator is *state merging*. It merges two states of a latent variable. Suppose a candidate model m' is obtained from m by merging two states, i and j, of a latent variable Z. Use $\hat{ij}$ to denote the merged state. Let $\bar{\mathcal{D}}_{\hat{ij}}$ be the data set obtained from $\bar{\mathcal{D}}$ by replacing both i and j with $\hat{ij}$. Then all the variables in m' are observed with respect to $\bar{\mathcal{D}}_{\hat{ij}}$. One can replace the first term of the BIC score by $\max_{\theta'} \log P(\bar{\mathcal{D}}_{\hat{ij}}|m', \theta')$ and use the resulting function to evaluate m'. This idea is borrowed from Elidan and Friedman (2005).

Now suppose m' is a candidate model in the first loop in the `simplify` subroutine, obtained from m by reducing the cardinality of a latent variable Z by 1. The DC method considers all possible ways to achieve the cardinality reduction through state merging, obtains a model score for each way using the above method, and uses the maximum of the scores to evaluate m'.

The candidate models in the `expand` subroutine are generated by two different operators. The DC method first separately evaluates candidate models generated by different operators, picks one model for each operator, runs full EM on the two resultant models and selects between them using improvement ratios.

Let m' be a candidate model obtained from the current model m by introducing a latent node Z to mediate a latent node Y and two

of its neighbors Z_1 and Z_2. Intuitively the new node would be necessary if, in the current model m, Z_1 and Z_2 are not conditionally independent given Y. This condition can be tested based on $\bar{\mathcal{D}}$ because all the three variables are observed in $\bar{\mathcal{D}}$. So the DC method evaluates m' using the G-squared statistic, computed from $\bar{\mathcal{D}}$ for testing the hypothesis that Z_1 and Z_2 are conditionally independent given Y. The larger the statistic, the further away Z_1 and Z_2 are from being independent given Y, and hence the more necessary it is to introduce the new node Z. This idea is due to Zhang and Kočka (2004).

Finally consider a candidate model m' obtained from the current model m by adding a new state to the domain of a latent variable Y. Let Z_1, Z_2, ..., Z_k be the neighbors of Y in m. Intuitively the new state would be necessary if the interactions among Z_1, Z_2, ..., Z_k are not properly modeled by Y. Let $\hat{P}$ be the empirical joint distribution computed from $\bar{\mathcal{D}}$ and P_m be the joint distribution given by the model m. For any two neighbors Z_i and Z_j of Y, the KL distance $KL(\hat{P}(Z_i, Z_j) \| P_m(Z_i, Z_j))$ is a measure of how well the interactions between them are modeled. The larger the KL distance, the poorer the interactions are modeled, and hence the more necessary it is to add a new state to Y. The DC method evaluates m' using the KL distance averaged over all pairs of neighbors of Y. This idea is due to Zhang and Kočka (2004).

Note that although the DC method conceptually starts with data completion, it does not compute an explicit representation of $\bar{\mathcal{D}}$. All the heuristic model scores can be computed from the current model (m, θ^*) and the original data set $\mathcal{D}$ and the computations all boil down to calculating sufficient statistics on some variables. Full EM is run on the model picked by any arg max operator to optimize its parameters before the search moves on to the next step.

5 Empirical Results

Coupling the EAST search procedure with either the RL method or the DC method for model evaluation, one obtains two different algorithms for learning LT models, which we denote by *EAST-RL* and *EAST-DC* respectively. The main purpose of our empirical studies is to compare the two algorithms.

We also attempt to answer two other questions. First, EAST-RL has two parameters μ and ν. How do they influence the performance of the algorithm? Second, the cost-effectiveness principle mentioned in section 2 was introduced to deal with the problem of operation granularity, i.e. some operations might increase model complexity much more than others. Is the principle necessary given that model complexity is considered already in the BIC score?

The Data: The synthetic data used in our experiments were generated using three manually constructed LT models that contain 7, 12 and 18 manifest variables respectively. Three data sets of sizes 1k, 5k and 10k were sampled from the 18 variable model. We denote them by $\mathcal{D}_{18}(1k)$, $\mathcal{D}_{18}(5k)$ and $\mathcal{D}_{18}(10k)$. One data set was sampled from each of the other two models. They consists of 5k and 10k samples and hence are denoted by $\mathcal{D}_7(5k)$ and $\mathcal{D}_{12}(10k)$. The data sets were analyzed by various algorithms and their variants. The quality of a learned model is measured by the empirical KL distance of the model to the corresponding generative model, an approximation to the true KL distance that was computed based on 5k testing data. The results are shown in Table 1. They are averaged over 10 runs. The standard deviations are given in the parentheses.

There are three real-world data sets. Their basic information is given in the table.

	# vars	# states per var	sample size train	sample size test
ICAC	31	3.5	1200	301
KIDNEY	35	4.0	2000	600
COIL	42	2.7	5822	4000

The COIL data set originates from the COIL Challenge 2000 (van der Putten and van Someren 2004). It consists of customer records of a Dutch insurance company. The ICAC data set is from a telephone survey by Hong Kong's anti-corruption agency on public perception about various issues related to corruption. KIDNEY is a medical data set studied by Zhang *et al.* (2008). Each of the data sets was split into two subsets, one for training and

Table 1: Empirical results on synthetic data (the top two) and on real-world data (the bottom).

	$\mathcal{D}_7(5k)$		$\mathcal{D}_{12}(10k)$	
	emp-KL	time(mins)	emp-KL	time(hrs)
EAST-RL(μ,ν) (1,20)	0.0117(2.1e-3)	3.8(2.2)	0.0062(4.2e-3)	1.8(0.3)
(8,20)	0.0105(1.3e-3)	5.9(1.6)	0.0049(3.8e-3)	2.2(0.3)
(8,40)	0.0101(4.5e-5)	7.1(0.1)	0.0032(2.4e-4)	2.6(0.2)
EAST-DC	0.0101(5.7e-5)	5.8(0.1)	0.0051(5.0e-3)	1.4(0.1)
EAST-RL0(μ, ν) (8,40)	0.0101(4.8e-05)	6.3(0.1)	0.0079(4.7e-3)	1.5(0.1)

	$\mathcal{D}_{18}(1k)$		$\mathcal{D}_{18}(5k)$		$\mathcal{D}_{18}(10k)$	
	emp-KL	time(hrs)	emp-KL	time(hrs)	emp-KL	time(hrs)
EAST-RL(μ,ν) (1,20)	0.1865(1.5e-5)	0.5(0.03)	0.0245(9.1e-3)	4.3(0.7)	0.0097(4.0e-3)	9.4(1.1)
(8,20)	0.1865(2.3e-5)	0.6(0.05)	0.0175(5.3e-3)	5.7(0.9)	0.0067(2.5e-3)	13.8(1.6)
(8,40)	0.1865(7.5e-6)	0.7(0.02)	0.0148(4.5e-3)	6.0(0.6)	0.0047(7.0e-4)	18.4(3.9)
EAST-DC	0.2171(3.3e-2)	0.6(0.04)	0.0371(3.5e-3)	3.9(0.4)	0.0113(3.0e-3)	8.2(1.5)
EAST-RL0(μ, ν) (8,40)	0.1865(6.3e-06)	0.6(0.01)	0.0326(1.1e-2)	4.4(0.9)	0.0207(1.2e-2)	10.1(1.8)

	KIDNEY			COIL			ICAC		
EAST-RL	BIC	logscore	time(days)	BIC	logscore	time(days)	BIC	logscore	time(days)
(μ,ν)=(4,10)	-57214(61)	-16882(41)	0.4(0.1)	-52116(205)	-34943(203)	0.9(0.2)	-26102(52)	-6198(23)	0.10(0.01)
(8,20)	-57158(73)	-16818(56)	0.6(0.1)	-51773(159)	-34577(195)	1.1(0.2)	-26033(22)	-6167(15)	0.16(0.03)
(16,40)	-57066(52)	-16761(25)	1.0(0.1)	-51505(74)	-34198(41)	2.3(0.4)	-26042(27)	-6173(15)	0.22(0.02)
EAST-DC	-57699(158)	-17236(156)	0.3(0.0)	-52560(295)	-35103(226)	0.7(0.1)	-26156(1)	-6213(13)	0.09(0.00)

the other for testing. LT models were obtained from the training sets. For each learned model, we computed its BIC score on the training set and its logarithmic score on testing set. The logscore measures how well the model predicts future data. The results are shown in Table 1. They are averaged over 5 runs.

Impact of μ and ν: The parameter μ controls the effort that local EM spends on avoiding local maxima, while ν controls the effort that local EM spends on refining parameters. In general we expect EAST-RL to find better models as they increase. Consider the KL distance from a learned model to the corresponding generative model as the parameter setting (μ, ν) changes from (1, 20) to (8, 20) and then to (8, 40). We see that the KL distance changes, on average, from 0.0097 to 0.0067 and then to 0.0047 for $\mathcal{D}_{18}(10k)$; from 0.0245 to 0.0175 and then to 0.0148 for $\mathcal{D}_{18}(5k)$; from 0.0062 to 0.0049 and then to 0.0032 for $\mathcal{D}_{12}(10k)$; and from 0.0117 to 0.0105 and then to 0.0101 for $\mathcal{D}_7(5k)$. It stays unchanged for $\mathcal{D}_{18}(1k)$.

The results on real-world data show similar trends with one exception. In the case of the ICAC data, the BIC score and the logscore drop slightly from (8, 20) to (16, 40), probably due to randomness.

EAST-RL vs. EAST-DC: The main purpose of our empirical studies is to compare EAST-RL and EAST-DC. Consider the experiments with synthetic data first. We compare the models found by the two algorithms in terms of the KL distances from those models to the corresponding generative models. For $\mathcal{D}_{18}(10k)$, the average KL distance of the models found by EAST-DC is 0.0113, while that of the models found by EAST-RL is 0.0097 in the lowest parameter setting and 0.0047 in the highest setting. As a matter of fact, EAST-RL found better models for $\mathcal{D}_{18}(10k)$ than EAST-DC in all the parameter settings considered. The same is true for $\mathcal{D}_{18}(5k)$ and $\mathcal{D}_{18}(1k)$. For $\mathcal{D}_{12}(10k)$, EAST-RL found better models than EAST-DC in all the settings except for (1, 20). For $\mathcal{D}_7(5k)$, EAST-RL found models of the same quality as EAST-DC in the highest setting.

EAST-RL also performed better than EAST-DC on the real-world data. It found better models on all the data sets and in all the settings.

Running times are also reported in Tables 1. They were collected on an Intel(R) Core(TM)2 PC with clock rate of 2.4GHz. EAST-DC is clearly more efficient than EAST-RL. On the COIL data, for instance, it was about 4 times faster than EAST-RL in the setting (16, 40).

Operator Granularity: In Table 1, EAST-RL0 denotes an implementation of EAST-RL where operation granularity is not considered when evaluating candidate models generated by SI and NI. We see that the models that EAST-RL0 found for $\mathcal{D}_{18}(10k)$, $\mathcal{D}_{18}(5k)$ and $\mathcal{D}_{12}(10k)$ are significantly worse than those found by EAST-RL. They are tied on the other

cases. This suggests that it is necessary to deal with operation granularity using the cost-effectiveness principle in the `expand` subroutine.

6 Discussions and Conclusions

This paper is concerned with the search-based approach to learning latent tree model. A key problem in the approach is how to efficiently evaluate large numbers of candidate models. A variety of ideas for attacking the problem were previously proposed by Zhang and Kočka (2004) and Elidan and Friedman (2005). In this paper we observe that those ideas can be grouped into two distinct approaches, namely the restricted likelihood (RL) approach and the data completion (DC) approach. We study and compare the approaches in the framework of EAST, a newly developed search procedure for learning latent tree models. This is the first time that the two approaches to efficient model evaluation are clearly identified and studied.

The RL method is conceptually simpler than the DC method. It is based on one principle, while the DC method is based on several heuristic ideas. The RL method is easier to understand than the DC method.

Ideally one should select the candidate model with the maximum BIC score. In the RL method, we replace the likelihood term in the BIC score with the maximum restricted loglikelihood. What it results in is an approximation that lower bounds the BIC score. So the RL method selects a model to maximize a lower bound of the true objective function. This is common practice in machine learning.

The DC method also selects candidate models to maximize some objective functions. However it is less clear how those functions are related to the BIC score. In the case of the `adjust` subroutine, some relationship exists because $\max_{\theta'} \log P(\bar{\mathcal{D}}|m', \theta') \geq \max_{\theta} \log P(\bar{\mathcal{D}}|m, \theta)$ implies $\max_{\theta'} \log P(\mathcal{D}|m', \theta') \geq \max_{\theta} \log P(\mathcal{D}|m, \theta)$. The same relationship is not known to be true for the other cases.

We empirically tested EAST-RL and EAST-DC on a number of synthetic and real-world data sets. Several parameter settings were tried for EAST-RL. At the lowest setting, EAST-RL found better models than EAST-DC on almost all the data sets and it took roughly the same amounts of time on almost all the data sets. At the highest setting, EAST-RL found better models than EAST-DC on all the data sets and much better models on many of the data sets. However, it was also significantly slower.

Acknowledgements

We thank Kin Man Poon for valuable discussions. Research on this work was supported by Hong Kong Grants Council Grants #622307, and China National Basic Research Program (aka the 973 Program) under project No.2003CB517106. The work was completed when the third author was on leave at the HKUST Fok Ying Tung Graduate School.

References

D. M. Chickering (2002). Learning Equivalence Classes of Bayesian-Network Structures. *JMLR*, 2.

D. M. Chickering and D. Heckerman (1997). Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison (1998) *Biological sequence analysis: probabilistic models of proteins and nucleic acids.* Cambridge Univ. Press.

G. Elidan and N. Friedman (2005). Learning hidden variable networks: the information bottleneck approach. *Journal of Machine Learning Research*, 6:81-127.

N. Friedman (1997). Learning belief networks in the presence of missing values and hidden variables. *ICML*.

D. Geiger, D. Heckerman and C. Meek (1996). Asymptotic Model Selection for Directed Networks with Hidden Variables. *UAI-96*, 158-168.

S. Guindon and O. Gascuel (2003). A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology*, 52(5):696-704.

P. F. Lazarsfeld and N. W. Henry (1968). *Latent structure analysis.* Houghton Mifflin, Boston.

I. Nachman, G. Elidan and N. Friedman (2004). "Ideal Parent" Structure Learning for Continuous Variable Networks. *UAI-04*, 400-409.

J. Pearl (1988). *Probabilistic reasoning in intelligence systems.* Morgan Kaufmann, San Mateo.

P. van der Putten and M. van Someren. A bias-variance analysis of a real world learning problem: The COIL challenge 2000. *Machine Learning*, 57:177-195.

N. L. Zhang (2004). Hierarchical latent class models for cluster analysis. *JMLR*, 5:697-723.

N. L. Zhang (2007). Discovery of Latent Structures: Experience with the CoIL Challenge 2000 Data Set. *ICCS-2007*, 26-34.

N. L. Zhang and T. Kočka (2004). Efficient learning of hierarchical latent class models. *ICTAI-2004*, 585-593.

N. L. Zhang, S. H. Yuan, T. Chen and Y. Wang (2008). Latent tree models and diagnosis in traditional Chinese medicine. *AI in Medicine*, 42:229-245.

Measuring Efficiency in Influence Diagram Models

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

Abstract

A measure of efficiency for influence diagram models with continuous decision variables that considers both the accuracy and complexity of the representation and solution technique is presented. Accuracy is determined by calculating the mean squared error between influence diagram decision rules and an analytical solution. Complexity is assessed by calculating the size of the functions in the numerical representation at each stage of the solution to reflect both storage requirements and potential computational complexity of downstream mathematical operations. The resulting efficiency score considers the preferences of an individual decision maker for accuracy and complexity. Three influence diagram models proposed for use with continuous decision variables are compared using the efficiency measurement. The best model for a given problem may vary based on a decision maker's willingness to substitute accuracy and complexity.

1 Introduction

The *influence diagram* (ID) was introduced by Howard and Matheson (1984) as a graphical and numerical representation for a decision problem under uncertainty. The ID model is composed of a directed acyclic graph that shows the relationships among chance and decision variables in the problem, as well as a set of conditional probability distributions for chance variables and a joint utility function. In addition to providing a tractable, intuitive view that facilitates communication about the decision problem, the ID solution provides an optimal strategy and maximum expected utility.

Since their invention, most subsequent improvements to exact solution procedures for solving IDs (see, e.g., citations in Cobb (2008)) assume that all chance and decision variables in the model are discrete. Cobb and Shenoy (2008) introduce mixtures of truncated exponentials (MTE) influence diagrams (MTEIDs), which are influence diagrams where probability density functions (pdfs) and utility functions are represented by MTE potentials (Moral et al. 2001). Each piece of an MTE function is composed of a sum of exponential terms where the exponent contains a linear function of the independent variables.

Discrete IDs and MTEIDs can only accomodate continuous decision variables if their state spaces are limited to a countable number of discrete values. In contrast, Shachter and Kenley (1989) introduce Gaussian IDs, where all continuous chance variables are normally distributed, all decision variables are continuous, and utility functions are quadratic. The mixture-of-Gaussians ID (Poland and Shachter 1993, Madsen and Jensen 2005) requires continuous chance variables to be modeled as mixtures of normal distributions and allows continuous decision variables.

Cobb (2007) introduces an ID model, the continuous decision MTE influence diagram (CDMTEID), which allows continuous decision variables with one continuous parent and continuous chance variables having any pdf.

The aim of this paper is to define a measurement that can be used to compare the efficiency of ID models with continuous decision variables.

This measurement captures both the accuracy and complexity of the ID solution, then weights these in accordance with the preferences of an individual decision maker. Using the efficiency metric, the competency in performance of ID solutions in three models—discrete IDs, MTEIDs, and CDMTEIDs—are compared. Using the measurements proposed in this paper, a decision maker can attempt to answer the question, "Is a more accurate model worth the additional computational complexity?"

The remainder of the paper is organized as follows. Section 2 describes notation and definitions. Section 3 provides details of the accuracy, complexity, and efficiency measurements. Section 4 provides an example of calculating accuracy and complexity. Section 5 compares efficiency results for the three ID models under consideration. Section 6 concludes the paper. This contribution is extracted from a longer working paper (Cobb 2008).

2 Notation and Definitions

2.1 Graphical Representation

Chance and decision variables in IDs are depicted as ovals and rectangles, respectively. Utility nodes appear as diamonds. An arrow pointing to a chance node indicates that the distribution for this chance node is conditional on the variable at the head of the arrow. An arrow pointing to a decision node means that the value of the variable at the head of the arrow will be known at the time the decision is made.

Example 1. Fig. 1 shows an ID model for a capacity planning and pricing decision problem under uncertainty (Gök 2002). Capacity (K) and price (P) are decision variables, the random demand "shock" (Z) is a chance variable, and u_0 is the joint utility function.

2.2 Numerical Representation

In this paper, we assume all decision and chance variables take values in bounded, continuous (non-countable) state spaces. All variables are denoted by capital letters in plain text, e.g., A, B, C, with sets of variables denoted by capital letters in boldface, e.g., $\mathbf{X}$. If A and $\mathbf{X}$ are

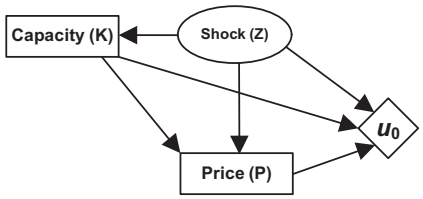


Figure 1: An influence diagram model.

one- and multi-dimensional variables, respectively, then a and $\mathbf{x}$ represent specific values of those variables.

The finite, continuous state space of $\mathbf{X}$ is denoted by $\Omega_{\mathbf{X}}$. The state space for a single variable B is defined as $\Omega_B = \{b : b_{min} \leq b \leq b_{max}\}$. At certain points in the ID representation and solution, a variable B's continuous state space, Ω_B, may be replaced by a discrete approximation, $\Omega_B^{(d)}$.

A probability potential, ϕ, for a set of variables $\mathbf{X}$ is a function $\phi : \Omega_{\mathbf{X}} \rightarrow [0, 1]$. A utility potential, u, for a set of variables $\mathbf{X}$ is a function $\Omega_{\mathbf{X}} \rightarrow \mathcal{R}$.

All piecewise functions are implicitly understood to equal zero in undefined regions.

Example 2. In the ID shown in Fig. 1, product demand is determined as $Q(p, z) = 12 - p + z$. Assume $Z \sim N(0, 1)$ and that the firm's utility (profit) function is

$$u_0(k, p, z) = \begin{cases} (p-1) \cdot (12 - p + z) - k \\ \qquad \text{if } (12 - p + z) \leq k \\ (p-1) \cdot k - k \\ \qquad \text{if } (12 - p + z) > k \,. \end{cases} \tag{1}$$

The state spaces of the variables are: $\Omega_K = \{k : 0 \leq k \leq 14\}$; $\Omega_P = \{p : 1 \leq p \leq 9\}$; and $\Omega_Z = \{z : -3 \leq z \leq 3\}$.

2.3 Combination

Combination of potentials is pointwise multiplication. Let ψ_1 and ψ_2 be probability and/or utility potentials for $\mathbf{X}_1$ and $\mathbf{X}_2$. The combination of ψ_1 and ψ_2 is a new potential for $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$ defined as

$$\psi(\mathbf{x}) = (\psi_1 \otimes \psi_2)(\mathbf{x}) = \psi_1(\mathbf{x}^{\downarrow \Omega_{\mathbf{X}_1}}) \cdot \psi_2(\mathbf{x}^{\downarrow \Omega_{\mathbf{X}_2}})$$

for all $\mathbf{x} \in \Omega_{\mathbf{X}}$. If either ψ_1 or ψ_2 is a utility potential, the result of the combination will be a utility potential; otherwise, the result is a probability potential.

2.4 Marginalization of Chance Variables

Marginalization of chance variables corresponds to integrating over the chance variable to be removed. Let ψ be a potential for $\mathbf{X} = \mathbf{X}' \cup Z$, where Z is a chance variable. The marginal of ψ for $\mathbf{X}'$ is a potential computed as

$$\psi^{\downarrow \mathbf{X}'}(\mathbf{x}') = \int_{\Omega_Z} \psi(\mathbf{x})\, dz$$

for all $\mathbf{x}' \in \Omega_{\mathbf{X}'}$, where $\mathbf{x} = (\mathbf{x}', z)$.

2.5 Marginalization of Decision Variables

Marginalization with respect to a decision variable is only defined for utility potentials. Let u be a utility potential for $\mathbf{X} \cup D$, where D is a decision variable. The marginal of u for $\mathbf{X}$ is a utility potential computed as

$$u^{\downarrow \mathbf{X}}(\mathbf{x}) = \max_{d \in \Omega_D} u(\mathbf{x}, d) \tag{2}$$

for all $\mathbf{x} \in \Omega_{\mathbf{X}}$. The mechanics of performing the maximization operation in Eq. (2) vary with each ID model compared in this paper.

2.6 Fusion Algorithm

IDs are solved in this paper by applying the fusion algorithm of Shenoy (1993). This algorithm involves deleting the variables in an elimination sequence that respects the information constraints in the problem. The sequence is chosen so that decision variables are eliminated before chance or decision variables that are immediate predecessors. When a variable is to be deleted from the model, all probability and/or utility potentials containing this variable in their domains are combined, then the variable is marginalized from the result.

3 Measuring Accuracy, Complexity, and Efficiency

3.1 Analytical Solution

In the problem from Examples 1 and 2, the firm knows the true value, $Z = z$, of the demand shock Z when it chooses capacity, so it would logically set $K = 12 - P + z$. Gök (2002) finds an analytical solution to the problem with optimal values for P and K of

$$p^* = \Theta_1^*(z) = 2 + \frac{10 + z}{2} \tag{3}$$

and

$$k^* = \Theta_2^*(z) = \frac{10 + z}{2}\ . \tag{4}$$

3.2 Accuracy

Since analytical decision rules are available for K and P, the mean squared error (MSE) (Winkler and Hays 1970) can be used as a measure of the difference between the analytical and ID decision rules. For instance, define Θ_2 as a decision rule for K as a function of Z determined using an ID method. The MSE of this function is calculated as

$$\begin{aligned} MSE &= E\left[(\Theta_2(z) - \Theta_2^*(z))^2\right] \\ &= \int_{\Omega_Z} \phi(z) \cdot (\Theta_2(z) - \Theta_2^*(z))^2\ dz\ , \end{aligned} \tag{5}$$

where ϕ is the pdf shown in Figure 2. The MSE between the decision rule Θ_1 developed in the ID models for P as a function of K and Z and the analytical decision rule Θ_1^* is similarly calculated as

$$\begin{aligned} MSE &= E\left[(\Theta_1(\Theta_2(z), z) - \Theta_1^*(z))^2\right] \\ &= \int_{\Omega_Z} \phi(z) \cdot (\Theta_1(\Theta_2(z), z) - \Theta_1^*(z))^2\ dz\ . \end{aligned} \tag{6}$$

The accuracy of a given ID model for this example will be denoted by $\mathcal{A}$ and will be defined as the sum of the MSEs calculated using Eqs. (5) and (6).

3.3 Complexity

The ID models in this paper are solved using Mathematica software (`www.wolfram.com`). This package provides a function called `LeafCount` that gives the total "size" of an expression defined using the `Piecewise` representation, based on applying the `FullForm` function (Wolfram 2003). `LeafCount` (denoted by $\mathcal{L}$) will be used to measure the complexity of potentials in the ID solution procedures.

Example 3. Consider the expression

$$f(z) = \begin{cases} -84.0 + 81.1 \exp\{0.0119(z+3)\} \\ \text{if } -3 \leq z \leq 3 \, . \end{cases} \tag{7}$$

Using the `Piecewise` environment, this function is defined in Mathematica as

```
f[z_] :=
  Piecewise{{-84.0
  +81.1 exp{0.0119(z + 3)}, -3 ≤ z ≤ 3}} .
```

Applying the `FullForm` function in Mathematica to this expression yields

```
Piecewise[List[List[ Plus[-84, Times[81.1,
  Power[E, Times[ 0.0119, Plus[3, z]]]]],
    LessEqual[-3, z, 3]]], 0] .
```

Each word, number, or variable in the `FullForm` expression increases the `LeafCount` of the expression by one. In this case, $\mathcal{L}\{f\} = 19$.

`LeafCount` captures the total size of all pieces of an MTE approximation or decision rule, including both the parameters of the function and the inequalities required to define the domains of each piece.

The complexity of the various ID methods will be determined by measuring the `LeafCount` of the potentials stored in memory after each combination or marginalization operation (or sub-operation thereof) performed in the solution technique. This measure of complexity is used because the size of the potentials at each step of the solution technique affects both the storage required and the subsequent number of calculations needed to solve the ID model.

Suppose the ID solution procedure for a particular problem requires n operations (or sub-operations) and denote the probability potentials, utility potentials, and decision rules remaining after operation i as ϕ_{ij}, $j = 1, \ldots, m_i$. The total complexity, $\mathcal{C}$, of the ID solution is the sum of the complexity measurements, $\mathcal{C}_i$, $i = 0, \ldots, n$, taken after each operation in the procedure. The total complexity of the solution for a particular ID model is determined as

$$\mathcal{C} = \sum_{i=0}^{n} \mathcal{C}_i = \sum_{i=0}^{n} \sum_{j=1}^{m_i} \mathcal{L}\{\phi_{ij}\} \, .$$

The value $\mathcal{C}_0$ represents the complexity of the potentials in the initial ID model.

3.4 Normalized Measurements

Since the MSE accuracy measurement, $\mathcal{A}$, and the complexity measurement determined by compiling `LeafCount` formulas, $\mathcal{C}$, are stated on different numerical scales, it is advantageous to normalize these two measurements onto a common scale to determine the trade-off between accuracy and complexity.

Select any two positive real numbers, $\mathcal{N}_{min}$ and $\mathcal{N}_{max}$. Throughout the remainder of the paper, we assume $\mathcal{N}_{min} = 1$ and $\mathcal{N}_{max} = 2$. When comparing the accuracy and complexity of ID solutions for multiple models, we denote by $\underline{\mathcal{A}}$ and $\underline{\mathcal{C}}$ the measurements for the least accurate and most complex models, respectively, measured for all models under consideration. Likewise, we denote by $\overline{\mathcal{A}}$ and $\overline{\mathcal{C}}$ the measurements for the most accurate and least complex models, respectively. In the case of both accuracy and complexity, note that smaller measurements are desirable. The normalized accuracy measurement for a given model is determined as

$$\hat{\mathcal{A}} = \mathcal{N}_{min} + \frac{(\mathcal{N}_{max} - \mathcal{N}_{min}) \cdot (\mathcal{A} - \underline{\mathcal{A}})}{\overline{\mathcal{A}} - \underline{\mathcal{A}}} \, . \tag{8}$$

Similarly, the normalized complexity measurement for a given model is calculated as

$$\hat{\mathcal{C}} = \mathcal{N}_{min} + \frac{(\mathcal{N}_{max} - \mathcal{N}_{min}) \cdot (\mathcal{C} - \underline{\mathcal{C}})}{\overline{\mathcal{C}} - \underline{\mathcal{C}}} \, . \tag{9}$$

3.5 Efficiency

Once the normalized accuracy and complexity measurements are determined, the *efficiency* of the model is calculated as

$$\mathcal{E} = \hat{\mathcal{A}}^{\alpha} \cdot \hat{\mathcal{C}}^{1-\alpha} \,, \tag{10}$$

The exponent α is a parameter assigned by the decision maker that conveys an individual preference for solutions that are either more accurate or less complex. If $\alpha > 0.5$, the decision maker values accuracy over complexity, and vice versa. Two properties of the functional form in Eq. (10) that make the expression a useful model for production and consumer utility in economics (see, e.g., Baye (2006)) also make it valuable for measuring the efficiency of ID solutions:

1. If two ID models have equivalent accuracy, the model with a better complexity score will have greater efficiency, and vice versa.

2. There is a diminishing marginal rate of substitution between accuracy and complexity.

4 Example

This section describes the calculation of the accuracy and efficiency measurements in the context of the CDMTEID solution to the problem from Examples 1 and 2. Details of calculations for the discrete ID and MTEID can be found in (Cobb 2008).

4.1 Representation

The MTE potential ϕ with $\mu = 0$ and $\sigma^2 = 1$ that approximates the normal distribution (as defined by Cobb and Shenoy (2006)) for the random demand shock (Z) is shown in Fig. 2, overlaid on the actual $N(0, 1)$ distribution.

The decision variable K is limited to discrete outcomes. A v-point discrete approximation to a continuous decision variable K with $\Omega_K = \{k : k_{min} \leq k \leq k_{max}\}$ has values $k_t = k_{min} + (t - 0.5) \cdot (k_{max} - k_{min})/v$ for $t = 1, \ldots, v$. Thus, the discrete state space is defined as $\Omega_K^{(k)} = \{k_1, k_2, \ldots, k_v\}$. To illustrate this example, we assume $v = 6$.

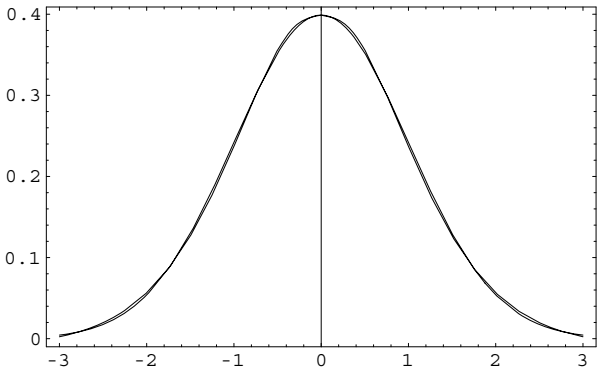


Figure 2: MTE probability density.

The function $f_1(p) = p$ on the interval $[1, 9]$ is modeled by the MTE potential $u_P(p) = -107.056144 + 108.102960 \exp\{0.0089234(p-1)\}$ for all $p \in \Omega_P$. Note that $u_P(1) = 1.047$, $u_P(5) = 4.975$, and $u_P(9) = 9.046$, so the MTE approximation fits $f_1(p)$ reasonably well. More accurate approximations can be obtained by dividing the state space of P and defining separate approximations over each region, at the expense of increasing the representation's complexity measurement (as defined in Section 3.3). The function $f_2(z) = z$ on $[-3, 3]$ is modeled with a similar approximation u_Z. With K assigned v discrete values, the MTE utility function is defined as

$$u_1(k_t, p, z) = \begin{cases} (u_P(p) - 1) \cdot \\ (12 - u_P(p) + u_Z(z)) - k_t \\ \qquad \text{if } (12 - p + z) \leq k_t \\ (u_P(p) - 1) \cdot k_t - k_t \\ \qquad \text{if } (12 - p + z) > k_t \,, \end{cases} \tag{11}$$

for $t = 1, \ldots, v$. For instance, with $v = 6$ and $K = k_3 = 5.83$, the MTE utility function is defined as

$$u_1(5.83, p, z) = \begin{cases} -3789.32 + 15328.9 \exp\{0.00892338p\} \\ -11479.5 \exp\{0.0178468p\} \\ +9002.5 \exp\{0.00892338p + 0.0118978z\} \\ -9079.3 \exp\{0.0118978z\} \\ \qquad \text{if } p - z \geq 6.17 \\ -636.2 + 625.0 \exp\{0.00892338p\} \\ \qquad \text{if } p - z < 6.17 \,. \end{cases}$$

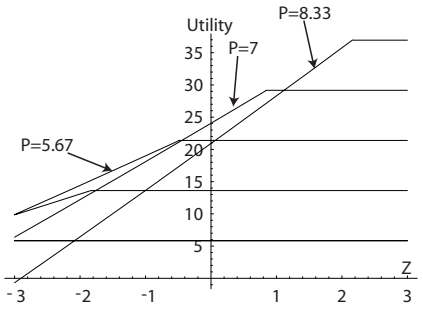


Figure 3: The utility functions $u_1(k_3, p_u, z)$ for $u = 1, \ldots, 6$.

The CDMTEID solution has initial complexity $\mathcal{C}_0 = \mathcal{L}\{\phi\} + \mathcal{L}\{u_1\} = 517$.

4.2 Solution

The elimination sequence employed in the fusion algorithm is P, K, Z.

Price (P) is a continuous decision variable; however, the first step in marginalizing this variable is accomplished by using the discrete approximation $\Omega_P^{(d)}$. The values p_u, $u = 1, \ldots, 6$ are inserted in the utility potential u_1 to form the utility functions $u_1(k_t, p_1, z)$, ..., $u_1(k_t, p_6, z)$ for $t = 1, \ldots, 6$. After this step, both these new potentials and the existing MTE utility function u_1 remain, so the complexity is

$$\mathcal{C}_1 = \mathcal{C}_0 + \sum_{t=1}^{6}\sum_{u=1}^{6} \mathcal{L}\{u_1(k_t, p_u, z)\} = 1313 \; .$$

The second step in removing P is to create a piecewise linear decision rule for P as a function of Z for each value k_t, $t = 1, \ldots, 6$. For $K = k_3 = 5.83$, the utility functions $u_1(k_3, p_u, z)$ for $u = 1, \ldots, 6$ are shown in Fig. 3 and we can conclude that $P = 5.67$ is optimal over $[-3, -0.45)$, $P = 7$ is optimal over $[-0.45, 1.15)$, and $P = 8.33$ is optimal over $[1.15, 3]$. These values are used to create the piecewise linear decision rule

$$P(z) = \Theta_{1,3}(z) =$$

$$\begin{cases} 6.775100 + 0.642570z & \text{if } -3 \leq z < -0.35 \\ 6.729469 + 0.772947z & \text{if } 0.35 \leq z < 2.9375 \\ 9 & \text{if } 2.9375 \leq z \leq 3 \; . \end{cases}$$

Similar decision rules $\Theta_{1,1}$, ..., $\Theta_{1,6}$ are determined corresponding to values k_t, $t = 1, \ldots, 6$ (see Fig. 4). When combined, these functions form the decision rule Θ_1 with complexity $\mathcal{L}\{\Theta_1\} = 259$. Since ϕ and u_1 also remain after this step, the complexity of the model is now $\mathcal{C}_2 = 517 + 259 = 776$.

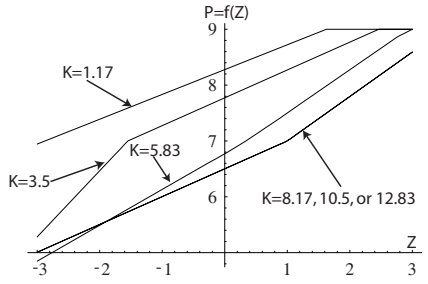


Figure 4: The piecewise linear decision rules $\Theta_{1,t}$ corresponding to values k_t, $t = 1, \ldots, 6$.

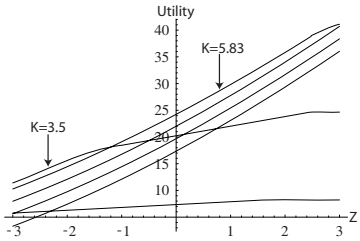


Figure 5: The utility functions $u_2(k_t, z)$.

The last step in removing P from the model is to substitute the values of the decision rule Θ_1 into the utility function u_1 to form the utility functions, $u_2(k_t, z) = u_1(k_t, \Theta_{1,t}(z), z)$, for $t = 1, \ldots, 6$. With ϕ and u_2 as the remaining potentials in the network, the complexity stands at

$$\mathcal{C}_3 = \mathcal{L}\{\phi\} + \sum_{t=1}^{6} \mathcal{L}\{u_2(k_t, z)\} = 61 + 530 = 591 \; .$$

A plot of the functions $u_2(k_1, z)$, ..., $u_2(k_6, z)$ (Fig. 5) shows that $u_2(3.5, z) \approx u_2(5.83, z)$ at $Z = -1.25$. The resulting decision rule Θ_2 specifies that $K = 3.5$ if $-3 \leq z < -1.25$ and $K = 5.83$ if $-1.25 \leq z \leq 3$. After creating this decision rule, the complexity of the model is

$$\mathcal{C}_4 = \mathcal{L}\{\phi\} + \mathcal{L}\{u_2\} + \mathcal{L}\{\Theta_2\} = 610 \; .$$

To complete the marginalization of K, we create a new utility function $u_3(z) = u_2\left(\Theta_2(z), z\right)$.

The complexity after this operation is $\mathcal{C}_5 = 278$, which captures the `LeafCount` of the potentials ϕ and u_3.

To remove Z, the potentials ϕ and u_3 are combined, with the resulting complexity $\mathcal{C}_6 = \mathcal{L}\{(\phi \otimes u_3)\} = 569$. Integrating the result over the state space of Z completes the solution. The total complexity of the ID model is

$$\mathcal{C} = \sum_{i=0}^{6} \mathcal{C}_i = 4654 \, .$$

The MSE of the CDMTEID solution is calculated according to Eqs. (5) and (6) as $\mathcal{A} = 0.7760$.

5 Results

This section discusses the effects on model efficiency of changing the number of states in the discrete approximations to continuous decision variables used in each of the methods

In each of the three ID methods illustrated, the state space of continuous variables is either permanently or temporarily discretized. To investigate the efficiency of models with a varying number of pieces in the discrete approximation, we consider the three ID models with approximations of six through twelve pieces. Thus, the best and worst solutions in terms of accuracy and complexity are chosen from among 21 models when calculating the values of $\underline{\mathcal{A}}$, $\overline{\mathcal{A}}$, $\underline{\mathcal{C}}$, and $\overline{\mathcal{C}}$.

Figs. 6 and 7 show efficiency scores for accuracy parameters of $\alpha = 0.1$ and $\alpha = 0.9$. When accuracy is a low priority, the efficiency of the models decreases with additional discrete pieces in the approximations as computational complexity overburdens the solution. In this case, both the CDMTEID and MTEID models provide comparable efficiency. When accuracy is a high priority, the efficiency of the models generally increases with additional pieces in the discrete approximations and the CDMTEID provides the best efficiency. In some cases, the placement of the mid-points of the discrete bins within the state space of the continuous decision variable adversely affects accuracy, which explains the low efficiency of the solutions with eight-piece approximations.

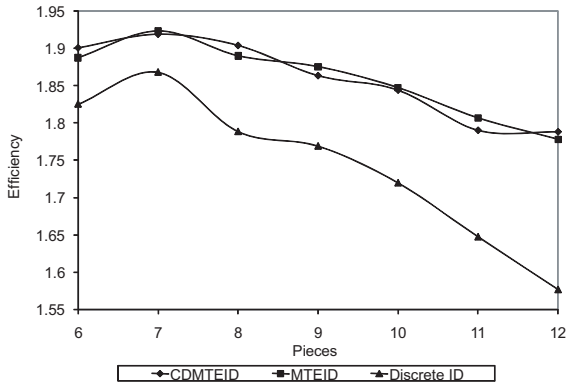


Figure 6: Efficiency scores with $\alpha = 0.1$.

Fig. 8 displays the efficiency scores for the ID solutions over the entire range of possible accuracy values. To make the graph simpler to comprehend, only the efficiency scores for models that gave the optimal efficiency over some range of the accuracy parameter α are shown. The un-normalized accuracy (MSE) and complexity values for these models are also displayed on the chart. For very low values of α, the MTEID solution with six discrete states is optimal. However, as the decision maker's preference for accuracy increases somewhat, a model with seven discrete states provides the best compromise between accuracy and complexity. Once desired accuracy increases beyond $\alpha \approx 0.25$, the CDMTEID model with an increasing number of discrete states in the temporary approximation

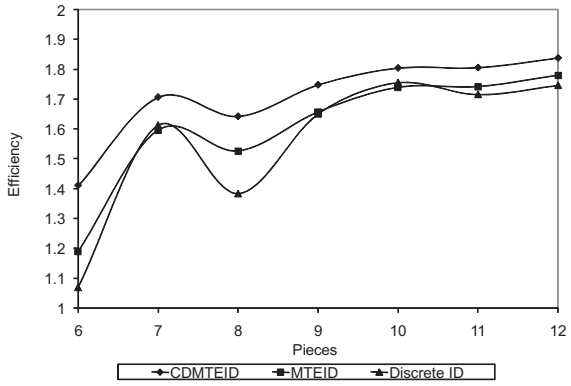


Figure 7: Efficiency scores with $\alpha = 0.9$.

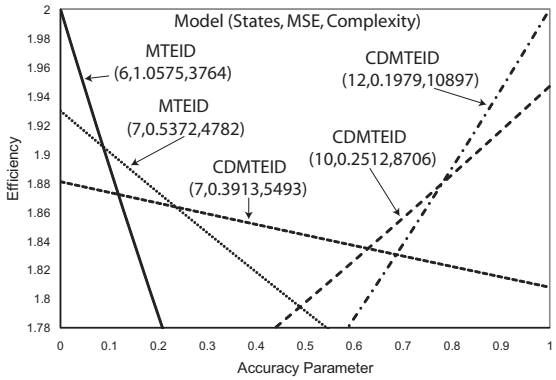


Figure 8: Efficiency values for ID solutions.

becomes optimal.

6 Conclusions

This paper has defined a measure of efficiency for ID models with continuous decision variables that considers both the accuracy and complexity of the representation and solution. Accuracy is measured by calculating the mean squared error between the final decision rule determined using the ID and corresponding analytical decision rules. Complexity is determined by the size of the potentials in the initial ID representation, and after each subsequent operation involved in solving the ID. The efficiency measurement combining accuracy and complexity is able to consider the preferences of an individual decision maker for both accuracy and complexity.

The details of additional comparisons and discussion of further results can be found in (Cobb 2008).

Acknowledgments

Support from the Spanish Ministry of Science and Innovation through project TIN2007–67418–C03–02 and from a Virginia Military Institute grant-in-aid is gratefully acknowledged.

References

Baye, M.R. 2006. *Managerial Economics and Business Strategy.* McGraw–Hill/Irwin, New York.

Cobb, B.R., 2007. Influence diagrams with continuous decision variables and non-Gaussian uncertainties. *Decision Anal.* **4**(3) 136–155.

Cobb, B.R., 2008. Efficiency of Influence Diagram Models with Continuous Decision Variables. Working Paper, Virginia Military Institute, Lexington, VA. Available for download at: `http://new.vmi.edu/media/ecbu/cobb/WP08b.pdf`

Cobb, B.R., P.P. Shenoy. 2006. Inference in hybrid Bayesian networks using mixtures of truncated exponentials. *Internat. J. Approx. Reason.* **41**(3) 257–286.

Cobb, B.R., P.P. Shenoy. 2008. Decision making with hybrid influence diagrams using mixtures of truncated exponentials. *Eur. J. Oper. Res.* **186**(1) 261–275.

Göx, R.F. 2002. Capacity planning and pricing under uncertainty. *J. Management Accounting Res.* **14**(1) 59–78.

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

Madsen, A.L., F. Jensen. 2005. Solving linear-quadratic conditional Gaussian influence diagrams. *Internat. J. Approx. Reason.* **38**(3) 263–282.

Moral, S., R. Rumí, A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. P. Besnard, S. Benferhart, eds. *Symbolic and Quantitative Approaches to Reasoning under Uncertainty: Lecture Notes in Artificial Intelligence*, Vol. 2143. Springer-Verlag, Heidelberg, 156–167.

Poland, W.B., R.D. Shachter. 1993. Mixtures of Gaussians and minimum relative entropy techniques for modeling continuous uncertainties. D. Heckerman, E.H. Mamdani, eds. *Uncertainty in Artificial Intelligence: Proc. Ninth Conf.*, Morgan Kaufmann, San Francisco, CA, 183–190.

Shachter, R.D., C.R Kenley. 1989. Gaussian influence diagrams. *Management Sci.* **35**(5) 527–550.

Shenoy, P.P. 1993. A new method for representing and solving Bayesian decision problems. D.J. Hand, ed. *Artificial Intelligence Frontiers in Statistics: AI and Statistics III.* Chapman and Hall, London, 119–138.

Winkler, R.L., W.L. Hays. 1970. *Statistics: Probability, Inference, and Decisions.* Holt, Rinehart, and Winston, New York.

Wolfram, S. 2003. *The Mathematica Book, 5th ed.* Wolfram Media, Champaign, IL.

Attribute Clustering Based on Heuristic Tree Partition

Jorge Cordero H. and Yifeng Zeng
Department of Computer Science
Aalborg University
9220 Aalborg, Denmark

Abstract

Attribute clustering has been previously employed to detect statistical dependence between subsets of variables. Clusters of variables can be appropriately used for detecting highly dependent domain variables and then reducing the complexity of learning Bayesian networks. We propose a novel attribute clustering algorithm motivated by research of complex networks, called the Star Discovery algorithm. The algorithm partitions and indirectly discards inconsistent edges from a maximum spanning tree by starting appropriate initial modes, therefore generating stable clusters. It discovers sound clusters through simple graph operations and achieves significant computational savings. We compare the Star Discovery algorithm against earlier attribute clustering algorithms and evaluate the performance in several domains.

1 Introduction

Probably one of the widest use of clustering in the past years has been the task of selecting genes (variable selection) in Bioinformatics. The use of attribute clustering can be extended to any domain in the search for statistical correlation of variables. Several conventional clustering algorithms have been applied to re-group and reveal subsets of correlated attributes such as: the k-means algorithm (Smet et al., 2002), fuzzy clustering (Madeira and Oliveira, 2004) and hierarchical clustering (Eisen et al., 1998).

Recently, the k-modes algorithm (Au et al., 2005) has been proved as one of the most efficient approaches for performing attribute clustering. However, it is subject to local optima due to random selection of initial modes. In a parallel line, clustering based on tree partition receives more and more attention since it is firmly rooted in classical graph partition methods (detailed methods will be presented soon in the next section). More precisely, the clustering methods firstly build a maximum spanning tree (MAST) and then get the clusters using appropriate partition methods. For convenience, we call the methods as MAST-based clustering algorithms in this paper. Since the standard tree partition method is not directly oriented toward attribute clustering it may not produce competitive results. However, it avoids heavy computation in contrast with k-modes algorithm. Accordingly, the MAST-based clustering algorithms contribute to the growing line of research on attribute clustering.

For the effect of this investigation we focus on the MAST-based clustering method. Specifically, we introduce the Star Discovery (SD) algorithm that is inspired by the research of complex networks (Cohen and Havlin, 2002). We adopt the assumption that all variables can be seen as points in an Euclidean space (close points have a high correlation) because we have complete information regarding pairwise proximities. The SD algorithm sections the tree by detecting nodes which have a strong connectivity; then, it pulls neighboring nodes into clusters based in a simple heuristic. We compare our approach against both earlier tree-based clustering algorithms and the k-modes algorithm in comprehensive experiments.

The rest of this paper is organized as follows: In Section 2 we present relevant algorithms for attribute clustering. Section 3 introduces the

novel SD algorithm. Section 4 exposes experimental findings. Section 5 provides a conclusive view of the work and discusses the use of attribute clustering for Bayesian networks.

2 Background

Given n domain attributes, $X = \{x_1, \cdots, x_n\}$, clustering methods aim to group a set of attributes[1] into clusters based on a similarity measure. In general, attributes in a cluster are more correlated to each other than to those ones belonging to different clusters. For this study, the data was statistically measured in terms of the interdependency redundancy measure $R(x_i, x_j) = \frac{I(x_i,x_j)}{H(x_i,x_j)}$; whereas $I(x_i, x_j) = \sum_{x_i,x_j \in X} p(x_i, x_j) \log \frac{P(x_i,x_j)}{p(x_i)p(x_j)}$ is the mutual information and $H(x_i, x_j) = \sum_{x_i,x_j \in X} p(x_i, x_j) \log p(x_i, x_j)$ is the joint entropy for the discrete random variables x_i and x_j (Au et al., 2005). The $R(\cdot, \cdot)$ measure discriminates a variable (containing many states) which has a weak statistical correlation with respect to another variable.

Without loss of generality, given a set of domain variables X, the objective of attribute clustering is to find a disjoint set of clusters $C = \{C_i | (i = 1, \cdots, k) \wedge (\forall_{i \neq j} C_i \cap C_j = \emptyset)\}$ that maximizes Eq. 1; where w_{o_i,x_j} denote the attached weight (measured by $R(o_i, x_j)$) from the center o_i to other variables x_j in the cluster C_i.

$$W^C = \sum_{C_i} \sum_{x_j \in (C_i - \{o_i\})} w_{o_i,x_j} \qquad (1)$$

Two paradigms of clustering were taken in order to find optimal clusters of discrete random variables. The first technique is the k-modes algorithm that optimize Eq. 1 directly (Au et al., 2005). The k-modes can be seen as a graph partitioning algorithm. Thus, a set of discrete random variables are exhibited as nodes in a complete graph ($K = (V, E)$, where V denotes a set of nodes representing variables X, and E includes all edges that are associated with all pair-wise $R(\cdot, \cdot)$ estimates). Another clustering method is the MAST-based clustering algorithm which partitions and clusters a tree instead of the complete graph. The overhead of constructing a maximum spanning tree is in the order of $O(n \log n)$ using the Kruskal's algorithm.

All of the clustering methods presented in this investigation input a set of weights $W^K = \{w_{x_i,x_j} = R(x_i, x_j) | i, j = 1, \cdots, n; i \neq j\}$ from the complete graph K.

2.1 The k-modes algorithm

The k-modes algorithm (also known as the k-medoids algorithm (Kaufman and Rousseeuw, 1990)) is basically an implementation of the k-means algorithm. It identifies the real points in the space as centers or modes rather than geometric centers. In fact, the k-modes is optimal in order to find well-shaped clusters since it has complete information among all pairwise interactions in the domain.

The k-modes algorithm works as follows: First, it initializes k random modes as cluster centers $O = \{o_1, \cdots, o_k\}$, and assigns every mode in a 1 to 1 correspondence to clusters C. Then, for every variable $x_j \in (X - O)$, it adds x_j to C_i iff $\forall_{o_l \in \{O - o_i\}} w_{x_j,o_i} > w_{x_j,o_l}$. Once the clusters C are constructed, a new variable $x_j \in C_i$ is selected as mode o_i in every cluster C_i iff $\sum_{x_j \in (C_i - \{o_i\})} w_{o_i,x_j}$ is maximal. The process is repeated (all clusters in C are deleted and a new set of clusters is created containing only the new modes) for a given number of iterations r or when no change in the modes is achieved. The complexity of this algorithm is polynomial $O(r(((n - k)k) + sk))$ where s is the maximal number of variables inside a cluster.

The k-modes algorithm is prone to falling into local optima due to the random mode selection in the initialization phase. A straightforward improvement could be done by feeding appropriate initial modes. We will show that our proposed algorithm may improve k-modes in this way.

[1] Discrete random variables (attributes) are seen as nodes in a graph ($V = X$, where V denotes a set of nodes). We will use any of these terms indifferently throughout this paper.

2.2 MAST Partitioning Algorithms

The MAST-based clustering algorithms are commonly based on heuristics that aim at removing a set of inconsistent edges from a MAST (Chow and Liu, 1968). An important factor in this technique is the selection of a heuristic or process that decides which arcs are relevant (and will remain) and which edges are inconsistent with the topology and shall be removed. These algorithms do not require many parameters to perform bisections over a tree. Moreover, this class of algorithms are faster than the k-means type of algorithms at the price of quality of the solution. We contemplated our study over three previous tree partitioning algorithms for attribute clustering as follows.

SEMST(The standard Euclidean maximum spanning tree (Asano et al., 1988)): The SEMST algorithm applies the principle of separability which states that two sets of points which are connected in a MST are separated by stabbing line. In other words, k sets of points can be isolated in a MAST if we remove the $k-1$ inconsistent edges whose weight is minimal.

By dividing the MAST G into k sub-trees, $G = \{G_1, \cdots, G_k\}$, the SEMST algorithm aims to maximize the sum of weights $W^G = \sum_{l=1}^{k} \sum_{x_i,x_j \in V_l} w_{x_i,x_j}$ where V_l is a set of variables in each sub-tree G_l. At the end, every set V_l becomes a cluster C_l.

The complexity of the SEMST algorithm is trivial since it takes $O(n \log n)$ to construct the Maximum Spanning Tree. If we use Kruskal's algorithm to build the initial MAST G then we already have sorted arcs according to their weights, in such case it will take constant time $O(k-1)$ to remove the inconsistent edges. For assembling of clusters it takes at most $O(kb)$ steps whereas b is the highest number of variables in a sub-tree G_l.

CEMST(The maximum cost spanning tree (Ye and Chao, 2004)): The algorithm works exactly as SEMST. However, the search for the k inconsistent edges is done by substituting the edge weights by the routing costs and then removing those edges that have maximal costs. A routing cost associated with an edge connecting the endpoints x_i and x_j is defined as: $Cost = w_{x_i,x_j} * Deg(x_i) * Deg(x_j)$, where $Deg(x_i)$ denotes the degree of x_i. Edges that connect leaf variables with the rest of the tree have higher probability of being discriminated since its own cardinality is low. The CEMST algorithm takes the same objective as that in the SEMST algorithm. Its complexity behaves in the same order as in the SEMST algorithm. Evidently, this algorithm as well as the SEMST algorithm do not directly optimize a specific objective function of attribute clustering. However, they indirectly aim to isolate clusters of highly related variables.

ZEMST(The Zahn's maximum spanning tree (Zahn, 1971)): Both the SEMST and CEMST algorithms perform a greedy blind search over the tree G in order to form clusters. In a parallel fashion, the ZEMST algorithm takes into account not only a given edge (x_i, x_j) but its relevance neighborhoods N_i, N_j respectively. A neighborhood $N_i = (V_{N_i}, E_{N_i})$ of a variable x_i in an edge (x_i, x_j), is a sub-tree that includes all reachable nodes V_{N_i} and arcs E_{N_i} of depth d (excluding paths starting from (x_i, x_j)).

In order to decide whether an edge (x_i, x_j) is inconsistent two tests are performed. First an attached weight w_{x_i,x_j} is removed if it is smaller than any of the means ($\bar{w}_{N_i} = \frac{1}{|E_{N_i}|} \sum_{(x_r,x_s) \in E_{N_i}} w_{x_r,x_s}$ and $\bar{w}_{N_j} = \frac{1}{|E_{N_j}|} \sum_{(x_t,x_u) \in E_{N_j}} w_{x_t,x_u}$) minus their standard deviations ($\sigma_{N_i} = (\frac{1}{|E_{N_i}|} \sum_{(x_r,x_s) \in E_{N_i}} (w_{x_r,x_s} - \bar{w}_{N_i}))^{\frac{1}{2}}$ and $\sigma_{N_j} = (\frac{1}{|E_{N_j}|} \sum_{(x_t,x_u) \in E_{N_j}} (w_{x_t,x_u} - \bar{w}_{N_j}))^{\frac{1}{2}}$) respectively. Second, all edges whose attached weight is higher than the mean in all the remaining sub-trees are removed.

Finally, the pruning process obtains the set of sub-trees $\{G_1, \cdots, G_k\}$. Every set of nodes in each sub-tree is mapped to a single cluster. Notice that the ZEMST algorithm automatically clusters the domain without receiving an initial number of partitions k. The complexity of this algorithm has to do with the search of neighborhoods among arcs. It has to perform a search of at most $d-1$ adjacent variables; thus, the

algorithm has a lower boundary in $O(nd)$ and a worst case scenario in $O(n^2)$ whenever $d \approx n$. The gathering of clusters is achieved (as in the previous algorithms) in a time $O(mb)$.

3 The Star Discovery Algorithm

We can intuitively realize that, as the rules for partitioning become more elaborated, then the final clustering has a better quality. Thus, the search for inconsistent edges is directed to isolate good clusters. In this section we introduce the robust Star Discovery (SD) algorithm. We iteratively partition a MAST and form clusters until all nodes $x_i \in X$ are assigned to clusters. The SD algorithm (as well as the ZEMST algorithm) clusters the domain in an unsupervised fashion (no initial number k of clusters is provided).

Guiding the search for centers by only examining the topology or single weights is probably not a good idea since the whole domain is not taken into account. The ZEMST algorithm bases the clustering in a simplistic search involving topology and weights in neighborhoods. We exploit further features in this way. A sound and clear approach is to look for subgraphs from the MAST that could reveal information about the "nature" of the domain. One abstraction of our technique is to look for spanning stars as subgraphs contained in the MAST. A spanning star (Gallian, 2007) is a sub-tree over the MAST, $S = (V_S, E_S)$, and is composed of q nodes. It has a center $o \in V_S$ with a degree $q-1$ and all other nodes have a degree of one. The spanning star is our fundamental graph theoretical resource for expressing clusters that reside in a two dimensional Euclidean space.

Detecting the set of k-stars whose global weight is maximal(following Eq. 1) from a complete graph K requires expensive computation. Similar to the previous MAST partitioning algorithms, the SD algorithm aims to detect a set of spanning stars, $SS = \{S_1, \cdots, S_k\}$, such that the objective function in Eq. 2 is maximized.

$$W = \sum_{S_l \in SS} \Big(\sum_{x_i \in Adj_l} (w_{x_i,o_l}) + \sum_{x_j \in Adj_l, x_h \in Leaf_l} (w_{x_j,x_h}) \Big) \quad (2)$$

where o_l is the star(cluster) center, Adj_l is a set of adjacent nodes to the center node o_l, and $Leaf_l$ a set of leaf nodes that connect to either o_l or Adj_l.

Notice that we extend the notion of a star to include some leaf nodes (nodes whose degree is 1 in the graph). In the experimentation we found that leaf nodes have a higher correlation to the center of its adjacent node than to any other center in any other star. The SD algorithm optimizes the later function by ranking every variable according to its ability to serve as modes. The search heuristic will only select a star as a mode if its mode has not been used before in any other clusters. At the end we will acquire the set of clusters whose structure (modes, adjacent and leaf nodes) is maximal according to Eq. 2 and the heuristic presented in Fig. 1 [2].

Star Discovery (SD) Algorithm

Input: $G = (V, E)$, W^G
Output: $C = \{C_1, C_2, \ldots, C_l\}$

1: $V^{aux} = V, V^{cont} = \emptyset, l = 1$
2: **FOR** $r = 1$ to n
3: $o_r = x_r$
4: $Adj_r \Leftarrow x_i$ **iff** $(x_i, o_r) \in E$
5: $E_{S_r} \Leftarrow (o_r, x_i)$
6: $Leaf_r \Leftarrow x_h$ **iff** $(x_i, x_h) \in E \wedge Deg(x_h) = 1$
7: $E_{S_r} \Leftarrow (x_i, x_h)$
8: $V_{S_r} = (o_r \cup Adj_r \cup Leaf_r)$
9: $S_r = (V_{S_r}, E_{S_r})$
10: $W^{S_r} = \sum_{(x_i,x_j) \in E_{S_r}} w_{x_i,x_j}$
11: $SS \Leftarrow S_r$
12: $W^{SS} \Leftarrow W^{S_r}$
13: Sort SS decreasingly according to W^{SS}
14: **WHILE** $V^{aux} \neq \emptyset$
15: $C_l = V_{S_l} - V^{cont}$
16: $V^{aux} = (V^{aux} - V_{S_l})$
17: $V^{cont} \Leftarrow V_{S_l}$
18: $C \Leftarrow C_l$
19: $l = l + 1$

Figure 1: The Star Discovery Algorithm.

The SD algorithm receives a MAST G and the set of weights W^G. At the very beginning the algorithm initializes an auxiliary set of variables V^{aux} and the counter l (line 1). After

[2] Note that $X \Leftarrow x$ indicates the addition of an element x to a given set X.

that, we build $n = |V|$ different stars, $S_r \in SS$, by specifying each variable x_r as the center o_r (line 3). For each star S_r, we include the adjacent nodes Adj_r to the center and leaf nodes $Leaf_r$ ($Deg(\cdot)$ denotes the node degree in the tree) (lines 4 and 6). Simultaneously, the edges are added (lines 5 and 7). Hence, the star S_r is a tuple having two sets: a set of nodes V_{S_r} and a set of edges E_{S_r} (line 9). In addition, we calculate the weight W^{S_r} in each star by adding all the weights attached to the star edges (line 10). Following, the auxiliary star S_r is kept in SS (line 11) as well as its corresponding weight W^{S_r} in W^{SS} (line 12).

Once the set of stars, SS, have been built from the MAST we proceed to sort them decreasingly in terms of the star weights (line 13). The sorting forms a ranking of potential modes and those ones with a higher weight W^{S_r} will be selected to form clusters (this way we form only one possible arrangement of clusters). We elect the star as the cluster C_l that has the largest star weight among the remained stars (line 15). We use V^{cont} to exclude variables already contained in previous clusters (line 17). This avoids possible overlapping nodes between any pair of clusters. A set of clusters C are completed until no nodes are left.

Assuming that there are n variables and the highest cardinalities of adjacent A_r and leaf L_r nodes are t and u respectively; then, the complexity in the first phase is $O(ntu)$ (lines 2-12) operations to search for all the adjacent nodes and leaves. The sorting operation takes at most $O(n \log n)$ if we use a merge-sort algorithm (line 13). The construction of clusters takes at most $O(l(t + u))$ operations (lines 14-19). Therefore the algorithm has a polynomial complexity $O((ntu)+(n \log n)+(l(t+u)))$. This polynomial complexity is better than the one in k-modes since the number of variables t and u is fairly low. Moreover, the SD algorithm is executed for a single time and not for a number of iterations as in the k-modes algorithm.

The SD algorithm always provides solutions that are deterministic. On the other hand, SD might not offer results that are better in quality than the ones given from the k-modes algorithm. However, k-modes could obtain better solutions in some cases, but it has the risk of falling into local optima (the solution depends of the initial modes).

4 Experimental Results

We discuss the reliability of the k-modes algorithm and then compare the performance of the SD algorithm against the aforementioned algorithms. A sound estimate to evaluate the goodness of a set of clusters uses Eq. 1. In other words, we are concerned to calculate the local degree of dependency between the centers or "modes" o_i of each cluster C_i against its other elements. Then, a global weight adds up every local weight in the clusters to obtain a total weight W^C.

For each experiment, we artificially generated datasets from some well known Bayesian networks such as: the Alarm (37 nodes), Barley (48 nodes), HeparII (70 nodes), Hailfinder (56 nodes) and Pathfinder (109 nodes) [3] (we abbreviated them as Al. Bar. Hep. Hai. and Pat. respectively). In this paper, we will only show the performance of the SD algorithm against earlier algorithms; a detailed discussion of some specific application of attribute clustering is subject to future work.

Reliability of the k-modes algorithm: Indeed, the k-modes algorithm can detect the optimal clustering given our objective. However, there is a drawback by using this approach. Since the formulation of the k-modes algorithm is greedy, there is the risk of falling into local optima. In order to test the susceptibility of the k-modes algorithm to fall into local optima, we fed initial modes ($k = 2$) in each domain with all the possible $\binom{n}{2}$ combinations of variables, then we ran the experiment until it converges. For this experiment, we generated a dataset for each domain with a sample size $\Omega = 10000$. Table 1 presents the results.

We found that k-modes does fall in local optima. For example, in the Alarm domain, it was interesting to see that k-modes converges into the optimal value of 6.13 with modes VentAlv and HR. However, it falls into 17 local optima

[3] http://genie.sis.pitt.edu/networks.html

Table 1: Number of local optima in which the k-modes algorithm falls.

Domains vs Local Optima			
Al.	Hep.	Hai	Path.
17	130	91	117

having modes (VentAlv, LVEDVolume), (VentAlv, Shunt), etc. In the optimal result, the size of the clusters is about $\frac{n}{2}$. In many local optima, one cluster becomes relatively small (10 variables). Clearly, a small cluster is isolated because of the sub-optimal initial mode. Whenever LVEDVolume or Shunt are selected as a mode, then no improvement is made. These modes dominate their neighborhoods. The previous analysis is a straightforward example of techniques based solely on an iterative greedy search. As shown in Table 1, the k-modes algorithm falls in more local optima values in larger domains. These findings are a strong motivation for developing an algorithm that could detect the right initial modes.

Clustering quality and sensitivity: We ran all of the algorithms SEMST (SE.), CESMT (CE.), ZEMST (ZE.), k-modes(k-m) and SD (using $k = 8$); then, we compared the quality of the clustering results in terms of its global weight W^C. For the effects of this experiment and to avoid local optima we fed the k-modes algorithm with the resulting modes of the SD algorithm (notice that we also fed k-modes with the final modes which were obtained by the other methods, but it fell into local optima). On the other hand, It is interesting to investigate the response of the clustering algorithms using different sample sizes (k was set to 8). As the sample size Ω decreases, the lectures of the $R(\cdot,\cdot)$ measure become less accurate. Depending on the domain (D) in study, there is a denominated level of sufficient statistics that determines the true nature of the MAST and reveals the true structure of correlated variables. Table 2 depicts the clustering results.

The SD algorithm performs better than the other tree-based clustering algorithms. Indeed, sometimes the SD algorithm is as effective as the k-modes algorithm. The later is true because if we consider the whole MAST in the cluster identification then we easily detect the strong components in the space. A highly connected variable in a MAST is very likely to be the best center in a given region. We can also conclude that more elaborated algorithms perform a better clustering. Clearly, the search spaces of the ZEMST and SD algorithms are relatively larger than the ones in the SEMST and CEMST approaches. Nevertheless, the search space of the SD algorithm is bigger than the one of ZEMST.

The SEMST, CEMST and ZEMST algorithms perform a *local* search on the MAST

Table 2: Performance(W^c) of the algorithms (Al.) in four domains over different sample sizes Ω. The k-modes algorithm is optimal when fed with the right initial modes.

		Ω			
D	**Alg.**	**10000**	**8000**	**6000**	**4000**
Al.	SE.	4.13	18.41	21.61	22.99
	CE.	5.4	18.78	22.07	23.52
	ZE.	6.11	19.10	22.85	24.66
	SD	7.85	**21.30**	**23.95**	**25.38**
	k-m	**8.35**	**21.30**	**23.95**	**25.38**
Bar.	SE.	2.33	14.67	19.03	22.23
	CE.	2.55	14.85	19.24	22.48
	ZE.	3.85	14.91	20.70	24.20
	SD	4.88	**15.39**	**21.02**	**25.41**
	k-m	**5.61**	**15.39**	**21.02**	**25.41**
Hep.	SE.	50.97	50.32	51.49	52.32
	CE.	51.21	50.55	51.71	52.89
	ZE.	51.27	51.43	52.55	53.54
	SD	**55.57**	**56.98**	**58.34**	**59.56**
	k-m	**55.57**	**56.98**	**58.34**	**59.56**
Hai.	SE.	30.26	31.33	32.42	33.65
	CE.	31.02	32.00	33.01	34.16
	ZE.	32.41	33.28	33.81	34.97
	SD	**32.48**	**33.58**	**34.69**	**35.96**
	k-m	**32.48**	**33.58**	**34.69**	**35.96**
Pat.	SE.	85.98	87.53	88.75	89.82
	CE.	88.63	88.22	89.40	90.19
	ZE.	88.315	88.75	89.64	90.61
	SD	86.61	89.31	89.71	91.03
	k-m	**90.33**	**89.41**	**91.32**	**92.72**

for clustering. For example, in the SEMST algorithm we completely disregard the inner relevance of an arc given the MAST topology. Thus, in practice, SEMST normally selects arcs connecting to leaf nodes as inconsistent (which in turn produces unbalanced bad clusters). In the CEMST algorithm, we take into account both weights and (up to some extent) the structure of the MAST. In this case, the inconsistent arcs have a maximal cost (which biases the search towards those arcs that are likely linked to highly connected nodes). The previous search technique is not enough since the search of inconsistent arcs is limited to a path of length 1. On the other hand, the ZEMST extends the search space by comparing the impact of removing an arc given some neighboring arcs and variables. Ultimately, the SD algorithm outperforms all the other tree-based algorithms because it calculates the clusters by considering both the weight and topology in the search. From the star formulation we realize that we could avoid local optima by discriminating those nodes that have a low connectivity and weight.

Conclusively, we can learn that the MAST is in fact a useful dependence graph whenever a sound clustering method is applied to section it. The same trend holds if we supply different sample sizes or change the number k of clusters.

We can see that all algorithms have the same behavior for different sample sizes. Clearly, the SD algorithm outperforms any other MAST-based clustering algorithms and obtains the same results as k-modes. Thus, the extensive search procedure of the SD algorithm secures competitive clustering.

Elapsed times: Finally, we investigated the running time of SD and other algorithms ($\Omega = 10000$). We used a system Centrino Duo with 2Ghz and 2 Gigabytes of memory. From Table 3 we can confirm that the algorithms calculate clusters obeying their complexity.

Logically, the SEMST algorithm is the fastest approach since it discards edges with the simplest rules. Ultimately, the elapsed times grow as the search space increases. The SD algorithm has a very competitive elapsed time (similar to the SEMST algorithm). We can see that in most cases, the SD clustering outperforms the k-modes algorithm in terms of elapsed times by a 50 percent ratio.

Table 3: Elapsed times (in seconds) for algorithms in all domains.

	D				
	Al.	Bar.	Hep.	Hai.	Pat.
SE	**0.031**	**0.04**	**0.044**	**0.049**	**0.047**
CE	0.04	0.042	0.056	0.05	0.062
ZE	0.078	0.057	0.065	0.07	0.094
SD	0.047	**0.04**	0.046	0.061	0.062
k-m	0.109	0.063	0.077	0.078	0.125

5 Discussion

In this paper, we illustrated a comprehensive study between several clustering algorithms. We found that the SD algorithm is able to obtain clusters having a better quality than using other MAST-based clustering algorithms. Hence, the SD algorithm can compete with the k-modes algorithm in some cases; the advantage of the SD algorithm over k-modes is that we obtain a single good solution. The SD algorithm can also be used to select the initial modes to be fed to the k-modes algorithm for further clustering. We aid the search of clusters by revealing the nature of the domain through a MAST structure. Therefore, the SD algorithm can be either used to perform the sectioning of a whole domain by itself, or to construct a hybrid algorithm (merged with the k-modes algorithm) which can find optimal clusterings. We also showed that our approach is straightforward to implement and fast to execute.

Attribute clustering is also relevant to the field of Bayesian networks. A cluster of attributes can be seen as a local set of variables in a large Bayesian network. Learning a large Bayesian network from data is still a difficult task since a large amount of computation is involved. Therefore, most learning algorithms adopt the divide and conquer strategy to alleviate the computational problem. These algorithms learn a large Bayesian net-

work by recovering small clusters of variables. For example, the Markov blanket is identified in the sparse candidate algorithm (Friedman et al., 1999) and the max-min hill climbing algorithm (Tsamardinos et a., 2006), the module framework in the learning module networks (Segal et al., 2003), and the block in the block learning algorithm (Zeng and Poh, 2004). The key feature in those approaches is the identification and union of components.

A component also represents reduced knowledge in the domain. For instance, some experts may be just interested in the specification of the left ulnaris or right ulnaris in the MUNIN network (Olensen et al., 1989) which consists of thousands of nodes. Attribute clustering in this case is a useful tool for variable selection in a massive domain. By performing this selection we may learn the structure of the desired variables in the domain or we could isolate only those important variables related to a target variable for study (this is useful because it helps us to visualize and focus on those relevant variables even when we have a tremendous amount of arcs in the network). Hence, the component formulation deserves further study.

Future work is in the search for true clustering applications. We may use the SD algorithm to discover knowledge in gene expression data. A more interesting application is to exploit the clustering algorithm for learning Bayesian networks. The key feature of such techniques will be the learning of large domains (with thousands of variables) by integrating small components into a full network.

References

M. K. T. Asano, B. Bhattacharya and F. Yao. 1988. Clustering algorithms based on minimum and maximum spanning trees. In *Proc. of the fourth annual symposium on Computational Geometry*, pages 252–257.

W. H. Au, K. Chan, A. Wong and Y. Wang. 2005. Attribute clustering for grouping, selection, and classification of gene expression data. *IEEE Trans. on Computational Biology and Bioinformatics*, 2(2):83–101.

C. Chow, and C. Liu. 1968. Approximating discrete probability distributions with dependence trees. In *IEEE/ACM Trans. on Information Theory*, 14(3):462–467.

D. B. Cohen and S. Havlin. 2004. *Structural Properties of Scale Free Networks*. Handbook of graphs and networks, Berlin GmbH: Wiley-Vch.

M. B. Eisen, P. T. Spellman, P.O. Brown and D. Botstein. Cluster Analysis and Display of Genome-Wide Expression Patterns. In *Proc. National Academy of Sciences of the United States of America*, 95(25):14863–14868.

N. Friedman, I. Nachman and D. Per. 1999. Learning Bayesian networks structure from massive dataset: The "sparse candidate" algorithm. In *UAI*, pages 206–215.

J. Gallian. 2007. Dynamic survey of graph labeling. In *Electronic Journal of Combinatorics*, 14(6).

L. Kaufman and P.J. Rousseeuw. 1990. Finding groups in data: An introduction to cluster analysis. John Wiley & Son.

S.C. Madeira and A.L. Oliveira. 2004. Biclustering Algorithms forn Biological Data Analysis: A Survey. *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, 1(1):24–45.

K. G. Olesen, U. Kjærulff, F. Jensen, F. V. Jensen, B. Falck, S. Andreassen and S. Andersen. A MUNIN network for the median nerve - A case study in loops. *Applied AI* 3: 385-404, 1989.

E. Segal, D. Peer, A. Regev, D. Koller, and N. Friedman. 2003. Learning module networks. In *Proc. of the 19th Conference on UAI*, pages 525–534.

F. De Smet, J. Mathys, K. Marchal, G. Thijs, B. DeMoor, and Y. Moreau. 2002. Adaptive Quality-Based Clustering of Gene Expression Profiles. *Bioinformatics*, 18(5):735–746.

I. Tsamardinos, L. E. Brown and C. F. Aliferis. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78.

B. Ye and K. M. Chao. 2004. *Spanning Trees and Optimization Problems*. Chapman and Hall.

C. Zahn. 1971. Graph theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. in Computers*, 20:68–86.

Y. F. Zeng and K.L. Poh. 2004. Block learning Bayesian network structures from data. In *Proc. of the Fourth International Conference on Hybrid Intelligent Systems*, pages 14–19.

A novel scalable and correct Markov boundary learning algorithm under faithfulness condition

Sergio Rodrigues de Morais
INSA-Lyon, LIESP, F-69622 Villeurbanne, France

Alex Aussem
University of Lyon 1, LIESP, F-69622 Villeurbanne, France

Abstract

In this paper, we propose a novel constraint-based Markov boundary discovery algorithm, called MBOR, that scales up to hundreds of thousands of variables. Its correctness under faithfulness condition is guaranteed. A thorough empiric evaluation of MBOR's robustness, efficiency and scalability is provided on synthetic databases involving thousands of variables. Our experimental results show a clear benefit in several situations: large Markov boundaries, weak associations and approximate functional dependencies among the variables.

1 Introduction

In this paper, we aim to identify the minimal subset of discrete random variables that is relevant for probabilistic classification in data sets with many variables but few instances (Guyon and Elisseeff, 2003). A principled solution to this problem is to determine the *Markov boundary* of the class variable T, i.e., the minimal subset of $\mathbf{U}$ (the full set), denoted by $\mathbf{MB}_T$ in the sequel, that renders the rest of $\mathbf{U}$ independent of T (Nilsson et al., 2007).

Following (Peña et al., 2007), we present a novel divide-and-conquer method, called MBOR, in order to increase the efficiency of the Markov boundary (MB for short) discovery while still being scalable and correct under the faithfulness condition. The problem with constraint-based algorithms is that the conditional independence tests become unreliable as the size of the conditional set increases. Such errors have usually a cascading effect that causes many errors in the final graph.

To get around this problem, MBOR combines rough and moderately accurate MB learners based on IAMB (Tsamardinos et al., 2003) and keeps the conditional test sizes of the tests as small as possible. A key difference between MBOR and all correct divide-and-conquer algorithms is the OR condition: two variables X and Y are considered as neighbors by MBOR if $Y \in PC_X$ OR $X \in PC_Y$, instead of the more stringent AND condition. Clearly, the OR condition makes it easier for true positive nodes to enter the Markov boundary, hence the name and the practical efficiency of our algorithm. Interestingly, some almost-deterministic relationships are also handled by the OR condition. The only difficulty was to maintain the correctness of the algorithm under the faithfulness condition.

In (Rodrigues de Morais and Aussem, 2008), we compared the ability of MBOR to solve real FSS problems using real data bases from the UCI Machine Learning Repository (e.g., Car Evaluation, Chess, Molecular Biology, SPECT heart, Tic-Tac-Toe, Wine and Waveform). In this study, we assess the scalability and the performance of MBOR through several experiments on synthetic databases with very few instances compared to the number of variables. MBOR is proved by extensive empirical simulations to be an excellent trade-off between running time and quality of reconstruction.

2 Notations and preliminaries

We denote a variable with an upper-case, X, and value of that variable by the same lower-case, x. We denote a set of variables by upper-case bold-face, $\mathbf{Z}$, and we use the corresponding lower-case bold-face, $\mathbf{z}$, to denote an assignment of value to each variable in the set. In this paper, we only deal with discrete random variables. We denote the conditional independence of the variable X and Y given $\mathbf{Z}$, in some distribution P by $X \perp_P Y|\mathbf{Z}$. Similarly, we write $X \perp_{\mathcal{G}} Y|\mathbf{Z}$ if X and Y are d-separated by $\mathbf{Z}$ in the DAG $\mathcal{G}$.

A Markov blanket $\mathbf{M}_T$ of the T is is any set of variables such that T is conditionally independent of all the remaining variables given $\mathbf{M}_T$. A Markov boundary, $\mathbf{MB}_T$, of T is any Markov blanket such that none of its proper subsets is a Markov blanket of T. In general, in a Baysesian network $< \mathcal{G}, P >$, we would want an edge to mean a direct dependency. As we know, the faithfulness entails this:

Definition 1. Suppose we have a joint probability distribution P of the random variables in some set $\mathbf{U}$ and a DAG $\mathcal{G} =< \mathbf{U}, \mathbf{E} >$. We say that $< \mathcal{G}, P >$ satisfies the faithfulness condition if, based on the Markov condition, G entails all and only conditional independencies in P.

Theorem 1. *Suppose $< \mathcal{G}, P >$ satisfies the faithfulness condition. Then for each variable X, the set of parents, children of X, and parents of children of X is the unique Markov boundary.*

A proof can be found for instance in (Neapolitan, 2004). A *spouse* of T is a another parent of a T's child node. We denote by $\mathbf{PC}_T$, the unique set of parents and children of T in $\mathcal{G}$ when $< \mathcal{G}, P >$, satisfies the faithfulness condition. Otherwise, $\mathbf{PC}_X^{\mathbf{U}}$ will denote the unique set of the variables that remains dependent on X conditioned on any set $\mathbf{Z} \in \mathbf{U} \setminus \{X, Y\}$.

3 Some problems with constraint-based methods

Constraint-based (CB for short) procedures systematically check the data for independence relationships to infer the structure. The association between two variables X and Y given a conditioning set $\mathbf{Z}$ is a measure of the strength of the dependence with respect to the data base $\mathcal{D}$. It is usually implemented with a statistical measure of association (e.g. χ^2, G^2). CB methods have the advantage of possessing clear stopping criteria and deterministic search procedures. On the other hand, they are prone to several instabilities: namely if a mistake is made early on in the search, it can lead to incorrect edges which may in turn lead to bad decisions in the future, which can lead to even more incorrect edges. This instability has the potential to cascade, creating many errors in the final graph (Dash and Druzdzel, 2003).

Insufficient data presents a lot of problems when working with statistical inference techniques like the independence test mentioned earlier. This occurs typically when the expected counts in the contingency table are small. The decision of accepting or rejecting the null hypothesis depends implicitly upon the degree of freedom which increases exponentially with the number of variables in the conditional set. So the larger the size of the conditioning test, the less accurate are the estimates of conditional probabilities and hence the less reliable are the independence tests. Another difficulty arises when true- or almost-deterministic relationships (ADR) are observed among the variables. Loosely speaking, a relationship is said to be almost deterministic when the fraction of tuples that violate the deterministic dependency is at most equal to some threshold. True DR are source of unfaithfulness but the existence of ADR among variables doesn't invalidate the faithfulness assumption. Several proposals have been discussed in the literature in order to reduce the cascading effect of early errors that causes many errors to be present in the final graph. The general idea is to keep the size of the conditional sets as small as possible in the curse of the learning process. Another idea is to reduce the degree of freedom of the statistical conditional independence test by some ways. The aim is twofold: to improve the data efficiency and to allow an early detection of ADR. Theses

strategies are not discussed here for conciseness, see (Yilmaz et al., 2002; Luo, 2006; Aussem et al., 2007; Rodrigues de Morais et al., 2008) for instance.

4 New method

In this section, we present in detail our learning algorithm called MBOR. We recall that MBOR was designed in order to endow the search procedure with the ability to: 1) handle efficiently data sets with thousands of variables but very few instances, 2) be correct under faithfulness condition, 3) handle implicitly some approximate deterministic relationships (ADR) without detecting them. We discuss next how we tackle each problem.

First of all, MBOR scales up to hundreds of thousands of variables in reasonable time because it searches the Markov boundary of the target without having to construct the whole Bayesian network first. Like PCMB (Peña et al., 2007) and MMMB (Tsamardinos et al., 2006), MBOR takes a divide-and-conquer approach that breaks the problem of identifying $\mathbf{MB}_T$ into two subproblems : first, identifying $\mathbf{PC}_T$ and, second, identifying the parents of the children (the spouses $\mathbf{SP}_T$) of T. According to Peña et al., this divide-and-conquer approach is supposed to be more data efficient than IAMB (Tsamardinos et al., 2003) and its variants, e.g., Fast-IAMB (Yaramakala, 2004) and Interleaved-IAMB (Yaramakala and Margaritis, 2005), because $\mathbf{MB}_T$ can be identified by conditioning on sets much smaller than those used by IAMB. Indeed, IAMB and its variants seek directly the minimal subset of $\mathbf{U}$ (the full set) that renders the rest of $\mathbf{U}$ independent of T, given $\mathbf{MB}_T$. Moreover, MBOR keeps the size of the conditional sets to the minimum possible without sacrificing the performance as discussed next.

The advantage of the divide-and-conquer strategy in terms of data efficiency does not come without some cost. MMMB (Tsamardinos et al., 2006) and PCMB (Peña et al., 2007) apply the "AND condition" to prove correctness under faithfulness condition. In other words, two variables X and Y are considered as neighbors if $Y \in PC_X$ AND $X \in PC_Y$. We believe this condition is far too severe and yields too many false negatives in the output. Instead, MBOR stands for "Markov Boundary search using the OR condition". This "OR condition" is a major difference between MBOR and all the above mentioned correct divide-and-conquer algorithms: two variables X and Y are considered as neighbors with MBOR if $Y \in PC_X$ OR $X \in PC_Y$. Clearly, the OR condition makes it easier for true positive nodes to enter the Markov boundary, hence the name and the practical efficiency of our algorithm. Moreover, the OR condition is a simple way to handle some ADR. For illustration, consider the sub-graph $X \Rightarrow T \rightarrow Y$, since $X \Rightarrow T$ is an ADR, $T \perp Y|X$ so Y will not be considered as a neighbor of T. As Y still sees T in its neighborhood, Y and T will be considered as adjacent by application of the OR condition. The main difficulty was to demonstrate the correctness under the faithfulness condition despite the OR condition.

MBOR (Algorithm 1) works in three steps and it is based on four subroutines called *PCSuperset*, *SPSuperset* and *MBtoPC* (Algorithms 2-4). Before we describe the algorithm step by step, we recall that the general idea underlying MBOR is to use a weak MB learner to create a stronger MB learner. By weak learner, we mean a simple and fast method that may produce many mistakes due to its data inefficiency. In other words, the proposed method aims at producing an accurate MB discovery algorithm by combining fast and moderately inaccurate (but correct) MB learners. The weak MB learner is used in *MBtoPC* (Algorithm 4) to implement a correct Parents and Children learning procedure. It works in two steps. First, the weak MB learner called *CorrectMB* is used at line 1 to output a candidate MB. *CorrectMB* may be implemented by any algorithm of the IAMB family because they don't implement the AND condition. In our implementation, we use Inter-IAMB for its simplicity and performance (Tsamardinos et al., 2003). The key difference between IAMB and Inter-IAMB is that the

shrinking phase is interleaved into the growing phase in Inter-IAMB. The second step (lines 3-6) of *MBtoPC* removes the spouses of the target.

In phase I, MBOR calls *PCSuperset* to extract **PCS**, a superset for the parents and children, and then calls *SPSuperset* to extract **SPS**, a superset for the target spouses (parents of children). Filtering reduces as much as possible the number of variables before proceeding to the MB discovery. In *PCSuperset* and *SPSuperset*, the size of the conditioning set $\mathbf{Z}$ in the tests is severely restricted: $card(\mathbf{Z}) \leq 1$ in *PCSuperset* (lines 3 and 10) and $card(\mathbf{Z}) \leq 2$ in *SPSuperset* (lines 5 and 11). As discussed before, conditioning on larger sets of variables would increase the risk of missing variables that are weakly associated to the target. It would also lessen the reliability of the independence tests. So the MB superset, **MBS** (line 3), is computed based on a scalable and highly data-efficient procedure. Moreover, the filtering phase is also a way to handle some ADR. For illustration, consider the sub-graph $Z \Rightarrow Y \rightarrow T \Leftarrow X$, since $X \Rightarrow T$ and $Z \Rightarrow Y$ are ADRs, $T \perp Y|X$ and $Y \perp T|Z$, Y would not be considered as a neighbor of T and vice-versa. The OR-condition in Phase II would not help in this particular case. Fortunately, as Phase I filters out variable Z, Y and T will be considered as adjacent

Phase II finds the parents and children in the restricted set of variables using the OR condition. Therefore, all variables that have T in their vicinity are included in $\mathbf{PC}_T$ (lines 7-8).

Phase III identifies the target's spouses in **MBS** in exactly the same way PCMB does (Peña et al., 2007). Note however that the OR condition is not applied in this last phase because it would not be possible to prove its correctness anymore.

The theorem below establishes MBOR's correctness under faithfulness condition:

Theorem 1. Under the assumptions that the independence tests are reliable and that the database is an independent and identically distributed sample from a probability distribution P faithful to a DAG $\mathcal{G}$, $MBOR(T)$ returns $\mathbf{MB}_T^{\mathbf{U}}$.

The proof may be found in (Rodrigues de Morais and Aussem, 2008). It is omitted here for conciseness. Note that the demonstration is not completely straightforward because a difficulty arises: as **MBS** is a subset of **U**, a marginal distribution $P^{\mathbf{V}}$ of $\mathbf{V} \subset \mathbf{U}$ may not satisfy the faithfulness condition with any DAG even if $P^{\mathbf{U}}$ does. This is an example of embedded faithfulness (Neapolitan, 2004) and every distribution doesn't admit an embedded faithful representation.

Algorithm 1 *MBOR*

Require: T: target; D: data set (**U** is the set of variables)
Ensure: [**PC**,**SP**]: Markov boundary of T

Phase I: *Find MB superset* (**MBS**)
1: $[\mathbf{PCS}, \mathbf{dSep}] = PCSuperSet(T, D)$
2: $\mathbf{SPS} = SPSuperSet(T, D, \mathbf{PCS}, \mathbf{dSep})$
3: $\mathbf{MBS} = \mathbf{PCS} \cup \mathbf{SPS}$
4: $\mathcal{D} = \mathcal{D}(\mathbf{MBS} \cup T)$ *i.e., remove from data set all variables in* $\mathbf{U}/\{\mathbf{MBS} \cup T\}$

Phase II: *Find parents and children of the target*
5: $\mathbf{PC} = MBtoPC(T, \mathcal{D})$
6: **for all** $X \in \mathbf{PCS} \setminus \mathbf{PC}$ **do**
7: **if** $T \in MBtoPC(X, \mathcal{D})$ **then**
8: $\mathbf{PC} = \mathbf{PC} \cup X$
9: **end if**
10: **end for**

Phase III: *Find spouses of the target*
11: $\mathbf{SP} = \emptyset$
12: **for all** $X \in \mathbf{PC}$ **do**
13: **for all** $Y \in MBtoPC(X, D) \setminus \{\mathbf{PC} \cup T\}$ **do**
14: Find minimal $\mathbf{Z} \subset \mathbf{MBS} \backslash \{T \cup Y\}$ such that $T \perp Y|\mathbf{Z}$
15: **if** $(T \not\perp Y|\mathbf{Z} \cup X)$ **then**
16: $\mathbf{SP} = \mathbf{SP} \cup Y$
17: **end if**
18: **end for**
19: **end for**

5 Experimental validation

In this section, we assess the scalability and the accuracy of MBOR through several experiments on synthetic databases with very few instances compared to the number of variables. We evaluate first the accuracy, the data-efficiency and running time of MBOR as the number of variables increases. Then, we compare the accuracy of MBOR against InterIAMB and PCMB

Algorithm 2 *PCSuperSet*

Require: T: target; D: data set (**U** is the set of variables)
Ensure: **PCS**: PC superset of T; **dSep**: d-separation set;

Phase I: *Remove X if $T \perp X$*
PCS $= \mathbf{U} \setminus T$
for all $X \in$ **PCS do**
  if $(T \perp X)$ **then**
    PCS $=$ **PCS** $\setminus X$
    dSep$(X) = \emptyset$
  end if
end for

Phase II:*Remove X if $T \perp X|Y$*
for all $X \in$ **PCS do**
  for all $Y \in$ **PCS** $\setminus X$ **do**
    if $(T \perp X \mid Y)$ **then**
      PCS $=$ **PCS** $\setminus X$
      dSep$(X) = Y$; **go to 15**
    end if
  end for
end for

Algorithm 3 *SPSuperSet*

Require: T: target; D: data set (**U** is the set of variables); **PCS**: PC superset of T; **dSep**: d-separation set;
Ensure: **SPS**: SP superset of T;

SPS $= \emptyset$
for all $X \in$ **PCS do**
  $\mathbf{SPS}_X = \emptyset$
  for all $Y \in \mathbf{U} \setminus \{T \cup \mathbf{PCS}\}$ **do**
    if $(T \not\perp Y|\mathbf{dSep}(Y) \cup X)$ **then**
      $\mathbf{SPS}_X = \mathbf{SPS}_X \cup Y$
    end if
  end for
  for all $Y \in \mathbf{SPS}_X$ **do**
    for all $Z \in \mathbf{SPS}_X \setminus Y$ **do**
      if $(T \perp Y|X \cup Z)$ **then**
        $\mathbf{SPS}_X = \mathbf{SPS}_X \setminus Y$; **go to 15**
      end if
    end for
  end for
  SPS $=$ **SPS** $\cup \mathbf{SPS}_X$
end for

Algorithm 4 *MBtoPC*

Require: T: target; D: data set
Ensure: **PC**: Parents and children of T;

MB $=$ *CorrectMB*(T, D)
PC $=$ **MB**
for all $X \in$ **MB do**
  if $\exists \mathbf{Z} \subset (\mathbf{MB} \setminus X)$ such that $T \perp X \mid \mathbf{Z}$ **then**
    PC $=$ **PC** $\setminus X$
  end if
end for

on six well-know BN benchmarks. To evaluate the accuracy, we combine precision (i.e., the number of true positives divided in the output by the number of nodes in the output) and recall (i.e., the number of true positives divided by the true size of the Markov Boundary) as $\sqrt{(1-precision)^2 + (1-recall)^2}$, to measure the Euclidean distance from perfect precision and recall, as proposed in (Peña et al., 2005). To implement the conditional independence test, we calculate the G^2 statistic as in (Spirtes et al., 2000), under the null hypothesis of the conditional independence. The significance level of the test is fixed to 0.05 for all algorithms. It might very well happen that several variables have the same association value with the target in data sets with very few instances. In this particular case, somewhat arbitrary (in)dependence decisions are taken. This can be seen as a source of randomness inherent to all CB procedures. To handle this problem, our implementation breaks ties at random: a random permutation of the variables is carried out before each algorithm is run.

5.1 Scalability

We compare first the accuracy of PCMB and MBOR through experiments on the INSURANCE (27 nodes/52 arcs) benchmark replicated several times (up to 1000 times) to increase the number of variables. We run MBOR and PCMB with the variable 'RiskAversion' as the target. The latter has 10 variables in its MB. Each network is obtained by tiling several copies of the initial INSURANCE network. The tiling is performed in a way that maintains the structural and probabilistic properties of the original network in the tiled network. We focus here on the accuracy and efficiency of the algorithms as a function of the number of variable in the tiled network (up to 27,000 variables). Clearly, the additional variables are all independent on the target. We report the number of conditional independence tests that were conducted (in log-log scale), the distribution of the conditioning test sizes and the Euclidean distance from perfect precision and recall, as a function of the number of variables in the tiled network. The

average and standard deviation values are estimated over 50 databases. As may be seen, MBOR requires fewer independence conditional tests than PCMB. The number of tests directly influences the execution time. It grows linearly with the number of nodes for both algorithms. The number of conditional tests of MBOR is 48% that of PCMB. As can be seen from Fig.1 (middle), while PCMB conducts (proportionally) fewer conditional tests which indicates improved test reliability, MBOR yields a significantly shorter distance in all cases Fig.1 (bottom).

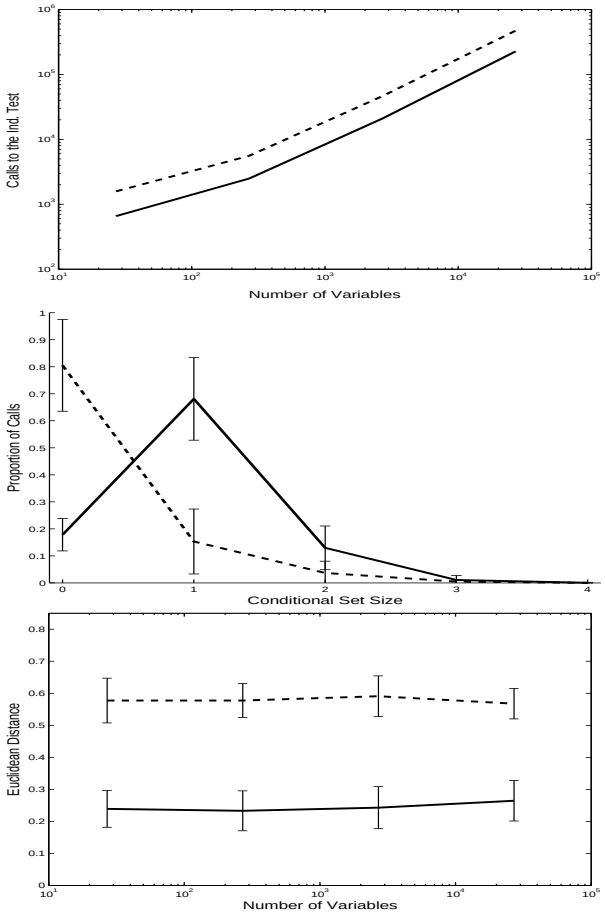


Figure 1: Insurance BN tiled several times. From top to bottom: number of conditional independence tests, distribution of the conditioning test sizes, Euclidean distance, as a function of the number of variables in the tiled BN. MBOR in plain line, PCMB in dotted line.

5.2 Accuracy

We report now the results of our experiments on six common benchmarks : BREAST-CANCER or ASIA (8 nodes/8 arcs), INSURANCE (27/52), INSULINE (35/52), ALARM (37/46), HAILFINDER (56/66) and CARPO (61/74). For each benchmark, we sampled 100 databases containing 100, 500 and 1000 instances respectively. The three algorithms were run, first, with each node in the BN as the target, and second, with the node with the largest MB in the BN as target. Figure 2 summarizes the empirical distribution of the Euclidean distance over 100 databases in the form of triplets of boxplots, one for each algorithm (PCMB, Inter-IAMB and MBOR respectively). Boxplots are convenient ways of graphically depicting the distributions of the Euclidean distances through their five-number summaries (the smallest observation, lower quartile, median, upper quartile, and largest observation). The boxplots also indicate (by the symbol '+') which observations, if any, might be considered outliers. In the left column of Fig.2, the distance is averaged over all nodes in the BN. In the right column, the distance for the node with the largest MB in the BN (i.e., ASIA : 'OR' (MB = 5 variables) ALARM : 'Intubation' & 'HR' (8 variables) INSULINE : 'IPA' & 'GPA' (18 variables) INSURANCE : 'RiskAversion' & 'Accident' (10 variables) HAILFINDER : 'CldShadeOth' (8 variables), CARPO : 'N69' (18 variables).

Several observations can be made from the results in Fig.2. First, it is rather surprising to observe that PCMB performs often worse than interIAMB even if PCMB is meant to conduct more reliable tests by conditioning on fewer variables. Despite the more reliable tests, the AND condition used in PCMB makes it hard for true positives to enter candidate MB. Second, the overall performance of MBOR and InterIAMB, when averaged over all nodes, is very similar (left column). For larger MBs, however, the advantages of MBOR against the other two algorithms are far more noticeable (right column). For instance, MBOR consistently outperforms the other algorithms, especially for

databases with 500 and 1000 instances. The larger the MB size, and the greater the gain in performance. As expected, the gain in accuracy is very significant on target variables 'IPA' & 'GPA' in INSULINE and on variable 'N69' in CARPO which contain 18 variables in their MB. The reason is that MBOR reduces drastically the average number of false negatives compared to PCMB and InterIAMB and this benefit comes at very little expense in terms of false positives. Moreover, the gain in accuracy seems to increase with the size of the database.

6 Conclusion

We discussed simple solutions to improve the efficiency of current constraint-based Markov boundary discovery algorithms. We proposed a novel approach called MBOR. Our experimental results on well-known benchmarks show a clear benefit in several situations: densely connected DAGs, weak associations or approximate functional dependencies among the variables.

References

A. Aussem, S. Rodrigues de Morais, and M. Corbex. 2007. Nasopharyngeal carcinoma data analysis with a novel bayesian network skeleton learning. In *11th Conference on Artificial Intelligence in Medicine AIME 07*, pages 326–330.

Denver Dash and Marek J. Druzdzel. 2003. Robust independence testing for constraint-based learning of causal structure. In *UAI*, pages 167–174.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

Wei Luo. 2006. Learning bayesian networks in semi-deterministic systems. In *Canadian Conference on AI*, pages 230–241.

R. E. Neapolitan. 2004. *Learning Bayesian Networks*. Prentice Hall.

R. Nilsson, J.M. Peña, J. Bjrkegren, and J. Tegnr. 2007. Consistent feature selection for pattern recognition in polynomial time. *Journal of Machine Learning Research*, 8:589–612.

J.M. Peña, J. Bjrkegren, and J. Tegnér. 2005. Scalable, efficient and correct learning of markov boundaries under the faithfulness assumption. In *8th European Conference on Symbolic and Quantitative Approaches to Reasoning under Uncertainty (ECSQARU 2005)*, volume 21, pages 136–147. Lecture Notes in Artificial Intelligence 3571.

J.M. Peña, R. Nilsson, J. Bjrkegren, and J. Tegnr. 2007. Towards scalable and data efficient learning of markov boundaries. *International Journal of Approximate Reasoning*, 45(2):211–232.

S. Rodrigues de Morais and A. Aussem. 2008. A novel scalable and data efficient feature subset selection algorithm. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD*, Antwerp, Belgium.

S. Rodrigues de Morais, A. Aussem, and M. Corbex. 2008. Handling almost-deterministic relationships in constraint-based bayesian network discovery : Application to cancer risk factor identification. In *16th European Symposium on Artificial Neural Networks ESANN'08*, pages 101–106.

Peter Spirtes, Clark Glymour, and Richard Scheines. 2000. *Causation, Prediction, and Search*. The MIT Press, 2 edition.

Ioannis Tsamardinos, Constantin F. Aliferis, and Alexander R. Statnikov. 2003. Algorithms for large scale markov blanket discovery. In *FLAIRS Conference*, pages 376–381.

Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. 2006. The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78.

Sandeep Yaramakala and Dimitris Margaritis. 2005. Speculative markov blanket discovery for optimal feature selection. In *ICDM*, pages 809–812.

Sandeep Yaramakala. 2004. Fast markov blanket discovery. In *MS-Thesis, Iowa State University*.

Yusuf Kenan Yilmaz, Ethem Alpaydin, H. Levent Akin, and Taner Bilgiç. 2002. Handling of deterministic relationships in constraint-based causal discovery. In *Probabilistic Graphical Models*.

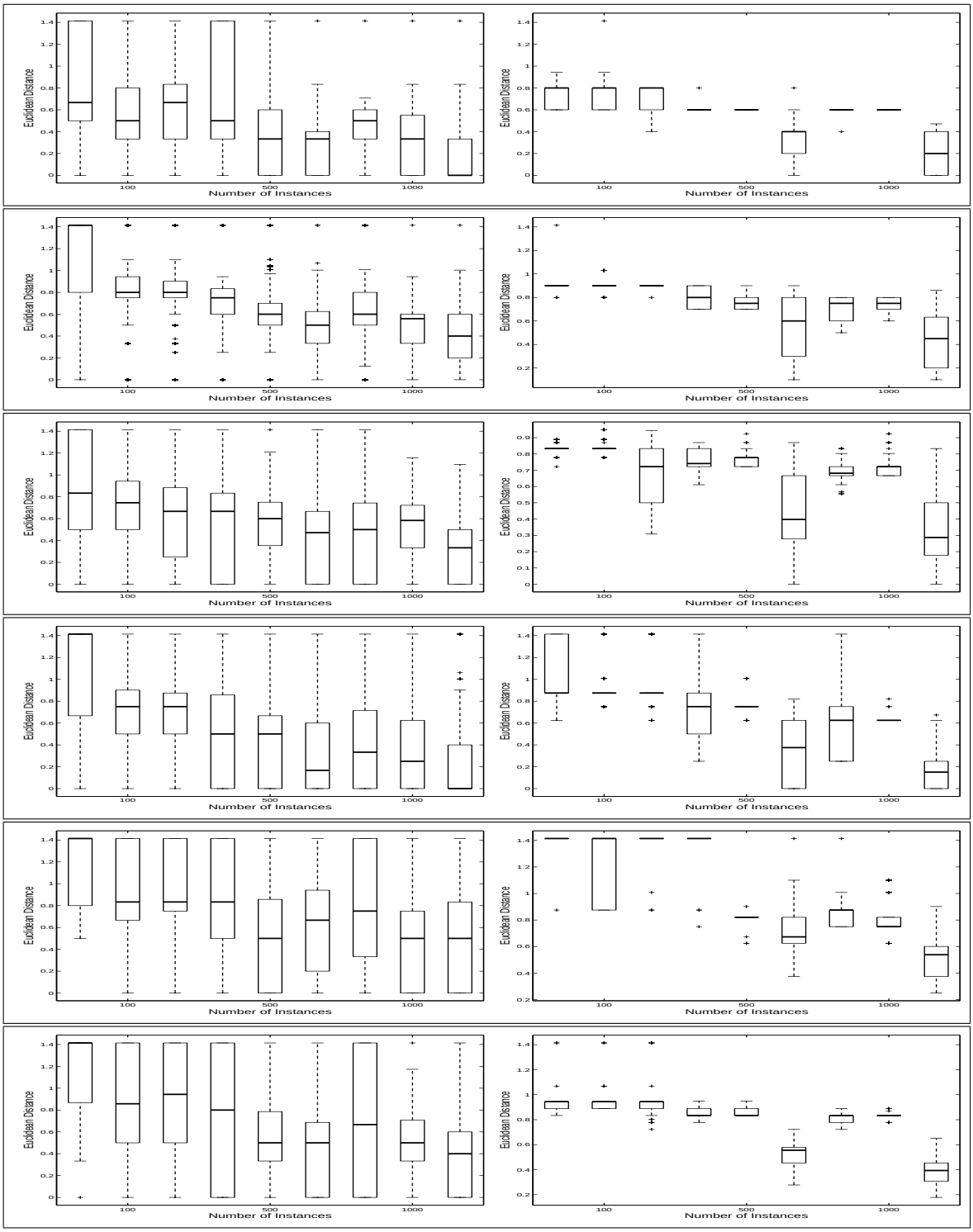


Figure 2: Empirical distribution of the Euclidean distance from perfect precision and recall over 100 databases. Results are shown for 100, 500 and 1000 instances in the form of triplets of boxplots for PCMB (left), InterIAMB (middle) and PCMB (right). From top to bottom: BREAST-CANCER, INSURANCE,, INSULINE, ALARM, HAIL-FINDER and CARPO. Left column: distance is averaged over all nodes in the BN. Right plot: distance for the node with the largest MB in the BN.

Marginals of DAG-Isomorphic Independence Models

Peter R. de Waal
Department of Information and Computing Sciences
Faculty of Sciences, Universiteit Utrecht

Abstract

Probabilistic and graphical independence models both satisfy the semi-graphoid axioms, but their respective modelling powers are not equal. For every graphical independence model that is represented by d-separation in a directed acyclic graph, there exists an isomorphic probabilistic independence model, i.e. it has exactly the same independence statements. The reverse does not hold, as there exists probability distributions for which there is no perfect map. We investigate if a given probabilistic independence model can be augmented with latent variables to a new independence model that is isomorphic with a graphical independence model of a directed acyclic graph. The original independence model can then be viewed as the marginal of the model with latent variables. We show that for some independence models we need infinitely many latent variables to accomplish this.

1 Introduction

Probabilistic models in artificial intelligence are typically built on the semi-graphoids axioms of independence. These axioms in fact are exploited explicitly in probabilistic graphical models, where independence is captured by topological properties, such as separation of vertices in an undirected graph or d-separation in a directed graph. A graphical representation with directed graphs for use in a decision support system has the advantage that it allows an intuitive interpretation by domain experts in terms of influences between the variables.

Ideally a probabilistic model is represented as a graphical model in a one-to-one way, that is, independence in the one representation implies independence in the other representation. The probabilistic model then is said to be isomorphic with the graphical model, and vice versa. Pearl and Paz (1987) established a set of sufficient and necessary conditions under which a probabilistic model is isomorphic with an undirected graph. In this paper we shall not consider representations of independence with undirected graphs, but focus on directed representations. Contrary to undirected graphs directed graphs allow the representation of induced dependencies: if a specific independence has been established given some evidence, it is possible that this independence becomes invalid if more evidence is obtained. Pearl (1988) gave a set of necessary conditions for directed graph isomorphism. To the best of our knowledge there is no known set of sufficient conditions.

Pearl (1988) also shows how a particular independence model that is not isomorphic with a directed graphical model, can be made isomorphic by the introduction of an auxiliary variable. In Pearl the isomorphism is then established by conditioning on the auxiliary variable. In this paper we choose a different approach. We extend the model with auxiliary variables to a directed graph isomorph and we then take the marginal over the original variables of this extended model. For this we introduce the concept of the marginal of a formal independence model. The model with auxiliary variables can then be considered as a latent perfect map. We show that it is possible to establish isomorphism in this manner, but that we may need an infinite number of auxiliary variables to accomplish this. We also show that there exists a probabilistic independence model that needs infinitely

many latent variables.

This paper is organised as follows. In Section 2 we briefly review probabilistic and graphical independence models, and the semi-graphoid properties of these models. In Section 3 we introduce the concept of marginals of an independence model and latent perfect maps. In Section 4 we discuss the existence of latent perfect maps, and in Section 5 we wrap up with conclusions and recommendations.

2 Preliminaries

In this section, we provide some preliminaries on probabilistic independence models as defined by conditional independence for probability distributions, graphical independence models as defined by d-separation in directed acyclic graphs, and formal independence models that capture the properties that probabilistic and graphical model have in common.

2.1 Conditional independence models

We consider a finite set of distinct symbols $V = \{V_1, \ldots, V_N\}$, called the *attributes* or *variable names*. With each variable V_i we associate a finite domain set $\mathcal{V}_i$, which is the set of possible values the variable can take. We define the domain of V as $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_N$, the Cartesian product of the domains of the individual variables.

A *probability measure* over V is defined by the domains $\mathcal{V}_i$, $i = 1, \ldots, N$, and a probability mapping $P : \mathcal{V} \to [0, 1]$ that satisfies the three basic axioms of probability theory (Kolmogorov, 1950).

For any subset $X = \{V_{i_1}, \ldots, V_{i_k}\} \subset V$, for some $k \geq 1$, we define the domain $\mathcal{X}$ of X as $\mathcal{X} = \mathcal{V}_{i_1} \times \cdots \mathcal{V}_{i_k}$. For a probability mapping P on V we define its *marginal* mapping over X, denoted by P^X, as

$$P^X(x) = \sum \Big\{ P(x, y) \,\Big|\, y \in \underset{\{i \,|\, V_i \notin X\}}{\times} \mathcal{V}_i \Big\}$$

for $x \in \mathcal{X}$. By definition $P^V \equiv P$ and $P^{\varnothing} \equiv 1$.

We denote the set of ordered triplets $(X, Y|Z)$ for disjoint subsets X, Y and Z of V as $\mathcal{T}(V)$. We shall use the notation $\mathcal{I}(X, Y|Z)$ to indicate $(X, Y|Z) \in \mathcal{I}$, for any ternary relation $\mathcal{I}$ on V. For simplicity of notation we will often write XY to denote the union $X \cup Y$, for $X, Y \subset V$. To avoid complicated notation we also allow Xy to denote $X \cup \{y\}$, for $X \subset V$ and $y \in V$.

Definition 1 (Conditional independence)**.** Let X, Y and Z be disjoint subsets of V, with domains $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, respectively. The sets X and Y are defined to be *conditionally independent under P given Z*, if for every $x \in \mathcal{X}$, $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$, we have

$$P^{XYZ}(x, y, z) \cdot P^Z(z) = P^{XZ}(x, z) \cdot P^{YZ}(y, z)$$

Definition 2. Let V be a set of variables and P a distribution on V. The *probabilistic independence model $\mathcal{I}_P$ of P* is defined as the ternary relation $\mathcal{I}_P$ on V for which $\mathcal{I}_P(X, Y|Z)$ if and only if X and Y are conditionally independent under P given Z.

If no ambiguity can arise we may omit the reference to the probability measure and just refer to the probabilistic independence model.

2.2 Graphical independence models in directed acyclic graphs

We first introduce the standard concepts of blocking and d-separation in directed graphs.

We consider a directed acyclic graph (DAG) $G = (V, A)$, with V the set of vertices and A the set of arcs. A path s in G of length $k - 1$ from a vertex V_{i_1} to V_{i_2} is a k-tuple $s = (W_1, W_2, \ldots, W_k)$ with $W_i \in V$ for $i = 1, \ldots, k$, $W_1 = V_{i_1}$, $W_k = V_{i_2}$ and for each $i = 1, \ldots, k - 1$ either $(W_i, W_{i+1}) \in A$ or $(W_{i+1}, W_i) \in A$. Without loss of generality we assume that a path has no loops, so there are no duplicates in $\{W_1, \ldots, W_k\}$. We define a path s to be *unidirectional* if all the arcs in s point in the same direction. More specifically, we define $s = (W_1, W_2, \ldots, W_k)$ to be a *descending* path if $(W_i, W_{i+1}) \in A$, for all $i = 1, \ldots, k - 1$. A descending path is unidirectional.

Definition 3. Let Z be a subset of V. We say that a path s is *blocked* in G by Z, if s contains three consecutive vertices W_{i-1}, W_i, and W_{i+1} for which one of the following conditions hold:

- $W_{i-1} \leftarrow W_i \rightarrow W_{i+1}$, and $W_i \in Z$,
- $W_{i-1} \rightarrow W_i \rightarrow W_{i+1}$, and $W_i \in Z$,
- $W_{i-1} \leftarrow W_i \leftarrow W_{i+1}$, and $W_i \in Z$,
- $W_{i-1} \rightarrow W_i \leftarrow W_{i+1}$, and $\sigma(W_i) \cap Z = \varnothing$, where $\sigma(W_i)$ consists of W_i and all its descendants.

We refer to the first three conditions as *blocking by presence*, and the last condition as *blocking by absence*. We refer to node W_i in the last condition as a *converging* or *colliding node* on the path.

While the concept of blocking is defined for a single path, the d-separation criterion applies to the set of all paths in G.

Definition 4. Let $G = (V, A)$ be a DAG, and let X, Y and Z be disjoint subsets of V. The set Z is said to *d-separate* X and Y in G, if every path s between any variable $x \in X$ to any variable $y \in Y$ is blocked in G by Z.

Based on the d-separation criterion we can define the notion of a graphical independence model.

Definition 5. Let $G = (V, A)$ be a DAG. The *graphical independence model* $\mathcal{I}_G$ *defined by* G is a ternary relation on V such that $\mathcal{I}_G(X, Y|Z)$ if and only if Z d-separates X and Y in G.

2.3 Formal independence models

Both a probabilistic independence model on a set of variables V and a graphical independence model on a DAG $G = (V, A)$ define a ternary relation on V. In fact we can capture this in a formal construct of an independence model.

Definition 6. A *formal independence model* on a set V is a ternary relation on V.

Both probabilistic and graphical independence models satisfy a set of axioms of independence. A special class within the set of formal independence models is defined based on these axioms.

Definition 7. A ternary relation $\mathcal{I}$ on V is a *semi-graphoid independence model*, or semi-graphoid for short, if it satisfies the following four axioms:

A1: $\mathcal{I}(X, Y|Z) \Rightarrow \mathcal{I}(Y, X|Z)$,

A2: $\mathcal{I}(X, YW|Z) \Rightarrow \mathcal{I}(X, Y|Z) \wedge \mathcal{I}(X, W|Z)$,

A3: $\mathcal{I}(X, YW|Z) \Rightarrow \mathcal{I}(X, Y|ZW)$,

A4: $\mathcal{I}(X, Y|Z) \wedge \mathcal{I}(X, W|ZY) \Rightarrow \mathcal{I}(X, YW|Z)$.

for all disjoint sets of variables $W, X, Y, Z \subset V$.

The axioms convey the idea that learning irrelevant information does not alter the relevance relationships among the other variables discerned. The four axioms are termed the *symmetry* (A1), *decomposition* (A2), *weak union* (A3) and the *contraction axiom* (A4), respectively.

The axioms were first introduced by Dawid (1979) for probabilistic conditional independence. These properties were later introduced in artificial intelligence as properties of separation in graphs by Pearl and Paz (1987) and Pearl (1988), and are since known as the *semi-graphoid* properties.

In the formulation that we have used so far we can allow X and Y to be empty, which leads to the so-called *trivial independence* axiom:

A0: $\mathcal{I}_P(X, \varnothing|Z)$,

This axiom trivially holds for both probabilistic independence and graphical independence.

An axiomatic representation allows us to derive qualitative statements about conditional independence, which may not be immediate from a numerical representation of probabilities. It also enables a parsimonious specification of an independence model, since it is sufficient to enumerate the so-called dominating independence statements, from which all other statements can be derived by application of the axioms (Studený, 1998).

2.4 Graph-isomorph

Since both probabilistic independence models and graphical independence models satisfy the semi-graphoid axioms, it is interesting to investigate whether they have equal modelling power. Can any probabilistic independence model also be represented by a graphical model, and vice versa? For this we introduce the notions of I-maps and P-maps.

Definition 8. Let $\mathcal{I}$ be an formal independence model on V, and $G = (V, A)$ a DAG that defines through d-separation a graphical independence model $\mathcal{I}_G$ on V.

1. The graph G is called an *independence map*, or *I-map* for short, for $\mathcal{I}$, if for all disjoint $X, Y, Z \subset V$ we have: $\mathcal{I}_G(X, Y|Z) \Rightarrow \mathcal{I}(X, Y|Z)$. If G is an I-map for $\mathcal{I}$, and deleting any arc makes G cease to be an I-map for $\mathcal{I}$, then G is called a *minimal I-map* for $\mathcal{I}$.

2. The graph G is called a *perfect map*, or *P-map* for short, for $\mathcal{I}$, if for all disjoint $X, Y, Z \subset V$ we have: $\mathcal{I}(X, Y|Z) \Leftrightarrow \mathcal{I}_G(X, Y|Z)$,

Definition 9 (DAG-isomorph)**.** An independence model $\mathcal{I}$ on V is said to be a *DAG-isomorph*, if there exists a graph $G = (V, A)$ that is a perfect map for $\mathcal{I}$.

Since a graphical independence model satisfies the semi-graphoid axioms, a DAG-isomorph has to be a semi-graphoid itself. Being a semi-graphoid is not a sufficient condition for DAG-isomorphism, however. To the best of our knowledge there does not exists a sufficient set of conditions, although Pearl (1988) presents a set of necessary conditions.

Some results from literature describe the modelling power of the independence models of the previous sections. Concerning the relationship between probabilistic and graphical models Geiger and Pearl (1990) show that for every DAG graphical model there exists a probability model for which that particular DAG is a perfect map. The reverse does not hold, there exists probability models for which there is no DAG perfect map (Pearl, 1988).

In (Studený, 1989) it is shown that the semi-graphoid axioms are not complete for probabilistic independence models. Studený derives a new axiom for probabilistic independence models that is not implied by the semi-graphoid axioms. He also shows in (Studený, 1992) that probabilistic independence models cannot be characterized by a finite set of inference rules.

3 Marginal of an isomorph

Pearl (1988) presents a set of necessary conditions for a formal independence model to be a DAG-isomorph. The conditions are based on properties of d-separation in DAG's. One of the conditions that is not already implied by the semi-graphoid axioms is the so-called *chordality condition*:

$$\mathcal{I}(x, y|zw) \wedge \mathcal{I}(z, w|xy) \Rightarrow \mathcal{I}(x, y|z) \vee \mathcal{I}(x, y, w)$$

for all $x, y, z, w \in V$. Pearl shows in (Pearl, 1988, Section 3.3.3) by example how conditioning on an auxiliary variable can be used to dispose of this chordality condition. In his example the independence model is not DAG-isomorph, but there exists a DAG with one extra variable, that, when conditioned on the auxiliary variable, is isomorphic with the independence model.

In this paper we choose a different approach where we introduce an auxiliary variable without conditioning to create a DAG that is a P-map for an independence model. We formulate this in the following definition.

Definition 10. Let $\mathcal{I}$ be an independence model on a set of variables V, and let A be a subset of V. We define the *marginal of $\mathcal{I}$ on A*, denoted by $\mathcal{I}^A$, as $\mathcal{I}^A = \mathcal{I} \cap \mathcal{T}(A)$.

We can now define a marginal version of DAG-isomorphism.

Definition 11 (DAG-isomorph marginal)**.** Let V be a set of variables, and $\mathcal{I}$ an independence model on V. We say that $\mathcal{I}$ is a *DAG-isomorph marginal*, if there exists a finite set of variables $\overline{V} \supseteq V$, an independence model $\overline{\mathcal{I}}$ on $\overline{V}$ and a DAG $\overline{G} = (\overline{V}, \overline{A})$, such that $\overline{G}$ is a P-map for $\overline{\mathcal{I}}$ and $\overline{\mathcal{I}}^V = \mathcal{I}$. We then say that $\overline{G}$ is a *latent P-map* of $\mathcal{I}$.

As an example we present the variable set $V = \{V_1, V_2, V_3, V_4\}$ and the formal independence model $\mathcal{I}$ on V defined by the following non-trivial independence statements (and their

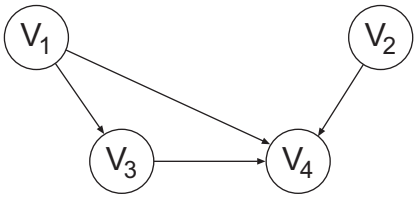


Figure 1: G_1, a minimal I-map for $\mathcal{I}$

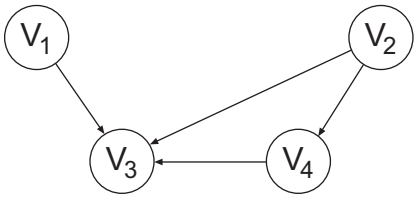


Figure 2: G_2, a minimal I-map for $\mathcal{I}$

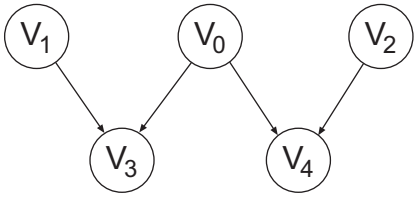


Figure 3: $\overline{G}$, a latent P-map for $\mathcal{I}$

symmetric equivalents):

$$\begin{array}{llll} (S1): & \mathcal{I}(V_1, V_2|\varnothing) & (S2): & \mathcal{I}(V_1, V_2|V_3) \\ (S3): & \mathcal{I}(V_2, V_3|\varnothing) & (S4): & \mathcal{I}(V_1, V_4|\varnothing) \\ (S5): & \mathcal{I}(V_1, V_2|V_4) & (S6): & \mathcal{I}(V_2, V_3|V_1) \\ (S7): & \mathcal{I}(V_1, V_4|V_2) & & \end{array}$$

The DAG $G_1 = (V, A)$ defined on the variables V as depicted in Figure 1, is a minimal I-map for $\mathcal{I}$, since the non-trivial graphical independence statement that can be derived from the DAG correspond to the statements (S1), (S2), (S3), and (S6). It is not a P-map for $\mathcal{I}$, since the statements (S4), (S5), and (S7) are not reflected as graphical independence statements in G_1. An alternative minimal I-map is G_2, as depicted in Figure 2. According to Dawid and Studený (1999, Lemma 5.1) there does not exist a P-map for $\mathcal{I}$ on V, although $\mathcal{I}$ satisfies the necessary conditions for DAG-isomorphism of Pearl (1988).

We can, however, construct a DAG $\overline{G}$ on a superset $\overline{V}$ of V for which the corresponding graphical independence model $\mathcal{I}_{\overline{G}}$ satisfies all the independence statements (S1)–(S7). This DAG is depicted in Figure 3. It has an extra, latent, variable V_0. The graphical independence model $\mathcal{I}_{\overline{G}}$ satisfies more independence statements than (S1)–(S7), like for instance $\mathcal{I}_{\overline{G}}(V_1, V_2|V_0)$. There are, however, no new independence statements $\mathcal{I}_{\overline{G}}(X, Y|Z)$ in $\mathcal{I}_{\overline{G}}$ for subsets X, Y, $Z \subset V$, other than (S1)–(S7). All new independence statements involve the latent variable V_0 in one of the arguments. By Definition 10 $\mathcal{I}$ in the example above is the marginal of $\mathcal{I}_{\overline{G}}$ on V, and $\overline{G}$ is a latent P-map of $\mathcal{I}$.

For the example we have from Geiger and Pearl (1990) that there exists a probability distribution $\overline{P}$ on $\overline{V}$ that has $\overline{G}$ of Figure 3 as a perfect map. The structure of $\overline{G}$ implies that $\overline{P}$ factorises as:

$$\begin{aligned} &\overline{P}(v_0, v_1, v_2, v_3, v_4) = \\ &\quad p_0(v_0)\, p_1(v_1)\, p_2(v_2)\, p_3(v_3|v_0v_1) p_4(v_4|v_0v_2) \end{aligned}$$

for some functions $p_1, \ldots, p_4$. It can be shown that the DAG's G_1 and G_2 are minimal I-maps for the marginal distribution of $\overline{P}$ on V. G_1 corresponds to a factorisation of P as:

$$\begin{aligned} &P(v_1, v_2, v_3, v_4) = \\ &\quad p_1(v_1)\, p_2(v_2)\, p'_3(v_3|v_1)\, p'_4(v_4|v_1v_2v_3) \end{aligned} \tag{1}$$

and G_2 corresponds to a factorisation of P as:

$$\begin{aligned} &P(v_1, v_2, v_3, v_4) = \\ &\quad p_1(v_1)\, p_2(v_2)\, p''_3(v_3|v_1v_2v_4)\, p''_4(v_4|v_2) \end{aligned} \tag{2}$$

In the example we thus have a probability distribution P and the corresponding independence model $\mathcal{I}_P$ on V that is not DAG-isomorphic, but it is the marginal of a distribution $\overline{P}$ that corresponds to a DAG-isomorphic probabilistic independence model.

For a probability measure we can now present a refined definition of DAG-isomorph marginal based on the probabilistic notion of a marginal.

Definition 12 (P-DAG-isomorph marginal)**.** Let V be a set of variables, P a probability measure on V. We say that P is a *P-DAG-isomorph marginal*, if there exists a finite set of variables $\overline{V} = \{\overline{V}_1, \ldots, \overline{V}_{\overline{N}}\} \supseteq V$ with domains $\overline{\mathcal{V}}_i$, $i = 1, \ldots, \overline{N}$, a DAG $\overline{G} = (\overline{V}, \overline{A})$, and a probability measure $\overline{P}$ on $\overline{V}$, such that

- The domains of the variables V_i in V for $\overline{P}$ are the same as for P ,
- The marginal distribution $\overline{P}^V$ of $\overline{P}$ over V is equal to P,
- $\overline{G}$ is a perfect map for $\overline{P}$.

The following lemma shows the relationship between Definitions 11 and 12.

Lemma 1. *Let V be a set of variables, P a probability measure with probabilistic independence model $\mathcal{I}_P$, and $\overline{P}$ a probability measure as in Definition 12. If we let $\mathcal{I}_{\overline{P}}$ denote the probabilistic independence model of $\overline{P}$, then $\mathcal{I}^V_{\overline{P}} = \mathcal{I}_P$.*

Proof. Let X, Y and Z be disjoint subsets of V, then the following holds:

$$\begin{aligned}
&\mathcal{I}_P(X, Y|Z) \\
\Leftrightarrow\ & P(X, Y, Z)P(Z) = P(X, Z)P(Y, Z) \\
\Leftrightarrow\ & \overline{P}^V(X, Y, Z)\overline{P}^V(Z) = \overline{P}^V(X, Z)\overline{P}^V(Y, Z) \\
\Leftrightarrow\ & \overline{P}(X, Y, Z)\overline{P}(Z) = \overline{P}(X, Z)\overline{P}(Y, Z) \\
\Leftrightarrow\ & \mathcal{I}_{\overline{P}}(X, Y|Z)
\end{aligned}$$

□

4 Existence of a latent perfect map

Weak transitivity and chordality are necessary conditions for DAG-isomorphism and therefore they must also be valid properties for DAG-isomorph marginals. This implies that any independence model that does not satisfy any of these two properties, is not a DAG-isomorph marginal. In the example of the previous section, which satisfies weak transitivity and chordality, we were able to construct a latent perfect map for the given independence model. In this section we show that a latent perfect map does not always exists, even if the independence model satisfies all Pearl's necessary conditions for a DAG-isomorph. The main result is captured in the following theorem.

Theorem 1. *There exists an independence model that satisfies the necessary conditions for DAG-isomorphism and has no latent P-map.*

We shall prove Theorem 1 by showing that there is no latent perfect map for the following independence model.

Definition 13. Let $V = \{B, C, D, E\}$ and let $\mathcal{I}^*$ be the independence model on V, that consists of the following three non-trivial independent statements (and their symmetric equivalents):

$$\begin{aligned}
(T1):&\ \ \mathcal{I}^*(B, E|CD) \\
(T2):&\ \ \mathcal{I}^*(C, E|\varnothing) \\
(T3):&\ \ \mathcal{I}^*(C, D|B)
\end{aligned}$$

It is a straight-forward exercise to verify that $\mathcal{I}^*$ is indeed a semi-graphoid. Application of the semi-graphoid axioms on (T1)–(T3) does not yield any new non-trivial independence statements. Moreover, $\mathcal{I}^*$ satisfies Pearl's necessary conditions for DAG-isomorphism.

We prove by contradiction that $\mathcal{I}^*$ is not a DAG-isomorph marginal. The steps in the proof are summarized in the following four lemmas.

Lemma 2. *Assume that $\mathcal{I}^*$, as defined in Definition 13, is a DAG-isomorph marginal and $\overline{G}$ is a latent P-map for $\mathcal{I}^*$, then there exists at least one path in $\overline{G}$ from C to E that is neither blocked by B nor by D.*

Proof. By contradiction: assume that there are no paths in $\overline{G}$ between C and E. C and E are then d-separated by any subset of $\overline{V}$, which contradicts, for instance, $\neg\mathcal{I}^*(C, E|BD)$.

Assume that all paths in $\overline{G}$ between C and E are blocked by B or D. Since there is at least one path in $\overline{G}$ from C to E, this again contradicts $\neg\mathcal{I}^*(C, E|BD)$. □

Lemma 3. *Assume that $\mathcal{I}^*$, as defined in Definition 13, is a DAG-isomorph marginal, $\overline{G}$ is a latent P-map for $\mathcal{I}^*$, and s is a path in $\overline{G}$ from C to E, then s has at least one converging node.*

Proof. Let s be a path from C to E. Due to $\mathcal{I}^*(C, E|\varnothing)$ s must be blocked by $\varnothing$, which implies that s has a converging node. □

Lemma 4. *Assume that $\mathcal{I}^*$, as defined in Definition 13, is a DAG-isomorph marginal, $\overline{G}$ is a latent P-map of $\mathcal{I}^*$, s a path in $\overline{G}$ from C to E that is neither blocked by B nor by D, and let F be a converging node on s, then $D \in \sigma(F)$ and $B \in \sigma(F)$. Moreover every descending path from F to D is blocked by B.*

Proof. If there exists a converging node F on s for which $B \notin \sigma(F)$ or $D \notin \sigma(F)$, then the path s would be blocked by B or D , which is in contradiction with the definition of s.

Let F be a converging node on s. Since $D \in \sigma(F)$, there exists a descending path s_1 from F to D. We now construct a new path s_2 from C to D by concatenating the subpath of s between C and F with s_1. Due to $\mathcal{I}^*(C, D|B)$ this path must be blocked by B. It cannot be blocked by B on the segment between C and F, since then also the original path s would be blocked by B. Therefore s_2 must be blocked by B on the subpath s_1. Since s_1 is descending, it is unidirectional. Therefore B must lie on s_1 and s_1 is blocked by B. □

Lemma 5. *Assume that $\mathcal{I}^*$, as defined in Definition 13, is a DAG-isomorph marginal, $\overline{G}$ is a latent P-map of $\mathcal{I}^*$, s is a path in $\overline{G}$ from C to E that is neither blocked by B nor by D. For any converging node F on s there is also a second converging node on the subpath of s between F and E.*

Proof. Let F be a converging node on s, which exists due to Lemma 3. From Lemma 4 we have that any descending path from F to D has B on it. At least one such path, say s_1, must exist, since $B \neq D$, so D cannot be equal to the converging node F. We now construct a path s_3 from B to E by concatenating the reverse of the part of subpath s_1 between B and F with the subpath of s between F and E (see also Figure 4).

Now s_3 is a path from B to E via F. Due to $\mathcal{I}^*(B, E|CD)$, this path s_3 must be blocked by CD. Since s_1 is descending and thus unidirectional, the first part of s_3 between B and F is unidirectional. D is not on this subpath, so it cannot be blocked by D. The second part of s_3 between F and E cannot be blocked by D, since it is part of the original path s and s is not blocked by D. In path s_3 the node F, where the two subpaths join, is not a converging node, so we conclude that s_3 cannot be blocked by D. This implies that s_3 must be blocked by C.

There are two possibilities for C to block s_3. The first possibility is that C blocks s_3 by presence on the (unidirectional) subpath s_1 between B and F. If this is the case, then we can construct a new path s_4 from C to E, by dropping from s_3 the first part between B and C. This new path s_4 consists of a unidirectional path between C and F, that has neither B nor D on it. The second part of the path, between F and E, is the segment of the original path s. Since F is not a converging node on s_4 and s is not blocked by B nor D, we conclude that s_4 is also not blocked by B nor by D. From Lemma 3 we conclude that s_4 must have a converging node, which can lie only between F and E. Therefore this converging node must also lie on the original path s.

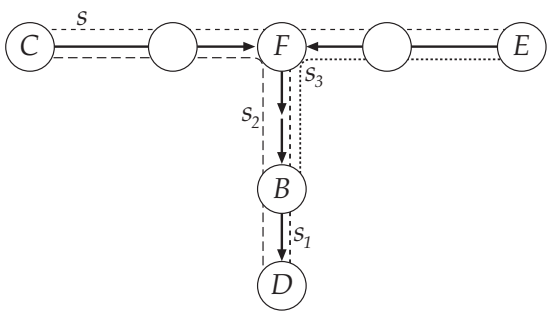


Figure 4: The paths used in the proofs of Lemmas 4 and 5

The second possibility for C to block s_3 is through absence, if there is a converging node on s_3 that does not have C as a descendant. Since the first part of s_3 between B and F is unidirectional, and F is not a converging node on s_3, this converging node must lie on the segment of s_3 strictly between F and E and therefore also on s. □

Proof of Theorem 1. Let $\mathcal{I}^*$ be as defined in Definition 13. Due to Lemma 2 we know that there is at least one path s between C and E that is not blocked by B nor by D. According to Lemma 3 this path s must have at least one converging node (Lemma 3) and due to Lemma 5 we can conclude that s must have an infinite number of converging nodes. Therefore $\overline{V}$ cannot be finite, and $\mathcal{I}^*$ is not a DAG-isomorph marginal. □

The next theorem shows that there is also a probabilistic independence model without a latent perfect map.

Theorem 2. *There exists a set of variables V and a probability distribution on V that is not a P-DAG-isomorph marginal.*

Proof. Consider the set of binary variables $V = \{B, C, D, E\}$. Define the probability measure P^* on V as follows:

B	C	D	E	$P^*(B,C,D,E)$
0	0	0	0	48/1357
0	0	0	1	48/1357
0	0	1	0	144/1357
0	0	1	1	48/1357
0	1	0	0	48/1357
0	1	0	1	96/1357
0	1	1	0	240/1357
0	1	1	1	48/1357
1	0	0	0	96/1357
1	0	0	1	96/1357
1	0	1	0	192/1357
1	0	1	1	64/1357
1	1	0	0	27/1357
1	1	0	1	54/1357
1	1	1	0	90/1357
1	1	1	1	18/1357

It can be verified that the probabilistic independence model $\mathcal{I}_{P^*}$ of P^* has exactly the same independence statements as $\mathcal{I}^*$ as defined in Definition 13. □

5 Conclusions

In this paper we have introduced the concept of the marginal of an formal independence model. We have shown that some independence models are in fact the marginals of models that are DAG-isomorphs, while the marginals themselves are not DAG-isomorphs. We have also proved that there exists some independence models for which we need to introduce an infinite number of auxiliary variables to obtain a latent perfect map. In examples for both cases the marginal independence models satisfy the sufficient conditions of Pearl (1988) for DAG-isomorphism. It is an interesting topic for future research to investigate if necessary and sufficient conditions can be established to guarantee the existence of a latent perfect map.

6 Acknowledgements

We would like to thank Richard D. Gill for suggesting this problem in the context of probabilistic graphical models, and an anonymous referee for valuable comments that helped to improve Sections 3 and 4.

References

A.P. Dawid. 1979. Conditional Independence in Statistical Theory, *Journal of the Royal Statistical Society B*, 41(1): 1–31.

A.P. Dawid and M. Studený. Conditional Products: An Alternative Approach to Conditional Independence, in *Artificial Intelligence and Statistics 99, Proceedings of the 7th workshop*, J. Whittaker and D. Heckerman (eds), Morgan Kaufmann, pp. 32-40.

D. Geiger and J. Pearl. 1990. On the Logic of Causal Models, in *Uncertainty in Artificial Intelligence 4*, R.D. Shachter, T.S. Levitt, L.N. Kanal, J.F. Lemmer (eds), Elsevier Science Publishers, 3–14.

A.N. Kolmogorov. 1950. *Foundation of the Theory of Probability*, New York, Chelsea Publishing.

J. Pearl, A. Paz. 1987. Graphoids, Graph-based Logic for Reasoning About Relevance Relations, in *Advances in Artificial Intelligence II*, B. du Boulay, D. Hogg, L. Steele (eds), 357–363.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems - Networks of Plausible Inference*, Morgan Kaufman.

M. Studený. 1989. Multi-information and the Problem of Characterization of Conditional Independence Relations, *Problems of Control and Information Theory*, 18 (1), 3–16.

M. Studený. 1992. Conditional Independence Relations Have No Finite Complete Characterization, in *Proc. of the 11th Prague conf. on Information Theory, Statistical Decision Functions and Random Processes*, S. Kubik, J.A. Visek (eds), 377–396.

M. Studený. (1998). Complexity of structural models. In: *Proceedings of the Joint Session of the 6th Prague Conference on Asymptotic Statistics and the 13th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, vol. II, Prague, pp. 521–528.

Policy Explanation in Factored Markov Decision Processes

Francisco Elizalde
Tec de Monterrey, Cuernavaca, Mexico
fef@iie.org.mx

L. Enrique Sucar
INAOE, Puebla, Mexico
esucar@inaoep.mx

Manuel Luque and Francisco Javier Díez
UNED, Madrid, Spain
{*mluque;fjdiez*}*@dia.uned.es*

Alberto Reyes
IIE, Cuernavaca, Mexico
areyes@iie.org.mx

Abstract

In this paper we address the problem of explaining the recommendations returned by a Markov decision process (MDP) that is part of an intelligent assistant for operator training. When analyzing the explanations provided by human experts, we observed that they concentrated on the "most relevant variable", i.e., the variable that in the current state of the system has the highest influence on the choice of the optimal action. We propose two heuristic rules for determining the most relevant variable based on a factored representation of an MDP. In the first one, we estimate the impact of each variable in the expected utility. The second rule evaluates the potential changes in the optimal action for each variable. We evaluated and compared each rule in the power plant domain, where we have a set of explanations, including the most relevant variable, given by a domain expert. Our experiments show a strong agreement between the variable selected by human experts and that selected by our method for a representative sample of states.

1 Introduction

Intelligent systems should be capable of explaining their decisions and reasoning process to the user. This is particularly important in the case of tutors and intelligent assistants. An important requirement for intelligent assistants is to have an explanation generation mechanism, so that the trainee has a better understanding of the recommended actions and can generalize them to similar situations (Herrmann et al., 1998).

Although there has been a lot of work in explanation generation for rule-based systems and other representations, there is very little work on explanations using probabilistic representations, in particular for decision–theoretic models such as influence diagrams and Markov decision processes (MDPs). We are particularly interested in explaining the recommendations obtained from an MDP that is part of an intelligent assistant for operator training. The assistant has a set of recommended actions (optimal policy) which compares to the ones performed by a person in a training session, and based on this gives advice to the user. In previous work (Elizalde et al., 2005) we used a set of predefined explanations produced by a domain expert, and these were given to the user according

to the current situation. A controlled user study showed that operators trained with the explanation mechanism have a better performance in similar situations (Elizalde et al., 2005). But obtaining the explanations from an expert is a complex and time-consuming process, so it is desirable that the assistant can generate the explanations automatically from the MDP and its solution.

When analyzing the explanations provided by human experts, we observed that they concentrated on the *most relevant variable*, i.e., the variable that in the current state of the system has the highest influence on the choice of the optimal action. That is, the expert's explanations start from certain aspect of the process that is the most important in the current situation and this aspect is the core of the explanation. So a first step towards automatic explanation based on MDPs is to determine the most relevant variable according to the current state and the optimal policy. The recommended action is also important for the explanation; however, this is directly obtained from the optimal policy that gives the solution of the MDP.

We have developed a novel technique for selecting the relevant variable for certain state-action based on a factored representation of an MDP. We propose two heuristic rules for obtaining the relevant variable, one based on utility and other based on policy. The utility–based rule evaluates how much the utility function will change if we vary the value of one of the variables for the current state, keeping the other variables fixed. The policy–based rule estimates the potential changes in optimal action for each of the variables. We compared the relevant variables obtained with these rules with the one given by the expert for a representative sample of states of an MDP in the domain of power plant operation. In general there was a strong agreement, which contributes evidence to the validity of the proposed approach.

The rest of the paper is organized as follows. Next we summarize related work on explanations based on probabilistic and decision-theoretic models. Then we present a brief review of MDPs. In section 4 we describe the proposed method for relevant variable selection. Experimental results are given in section 5, where we describe the test domain and the intelligent assistant. We conclude with a summary and directions for future work.

2 Related Work

The work on explanations based on probabilistic graphical models (PGMs) can be divided according to the classes of models considered, basically Bayesian networks (BN's) and decision networks. BN's (Pearl, 1988) graphically represent the dependencies of a set of random variables, and are usually used for estimating the posterior probability of some variables given another. So the main goal of explanations is to try to understand this inference process, and how it propagates through the network. Two main strategies have been proposed for explanation with BN's. One strategy is based on transforming the network to a qualitative representation, and using this more abstract model to explain the relations between variables and the inference process (Druzdzel, 1991), (Renooij and van der Gaag, 1998). The other strategy is based on the graphical representation of the model, using visual attributes (such as colors, line widths, etc.) to explain relations between nodes (variables) as well as the the inference process (Lacave et al., 2000). The explanation of links represents qualitative influences (Wellman, 1990) by coloring the links depending on the kind of influence transmitted from its tail to its head. Another possibility for static explanation consists of explaining the whole network.

Influence diagrams extend BNs by incorporating decision nodes and utility nodes. The main objective of these models is to help in the decision making process, by obtaining the decisions that maximize the expected utility. So explanation in this case has to do with understanding why some decision (or sequence of decisions) is optimal given the current evidence. There is very little work on explanations for decision networks. Bielza et al. (2003) propose an explanation method for medical expert systems based on influence diagrams. It is based

on reducing the table of optimal decisions obtained from an influence diagram, building a list that clusters sets of variable instances with the same decision. They propose to use this compact representation of the decision table as a form of explanation, showing the variables that are fixed as a rule for certain case. It seems like a very limited form of explanation, difficult to apply to other domains. The explanation facilities for Bayesian networks proposed by Lacave et al. (2000) were extended to influence diagrams and integrated in the Elvira software (Lacave et al., 2007). The extension is based in a transformation of the influence diagram into a Bayesian network by using a strategy for the decisions in the model. Lacave et al. (2007) describe several facilities: incorporating evidence into the model, the conversion of the influence diagram into a decision tree, the possibility of analyzing non-optimal policies imposed by the user, and sensitivity analysis with respect to the parameters.

Markov decision processes can be seen as an extension of decision networks, that consider a series of decisions in time (dynamic decision network). Some factored recommendation systems use algorithms to reduce the size of the state space (Givan et al., 2003) and perform symbolic manipulations required to group similarly behaving states as a preprocessing step. (Dean and Givan, 1997) also consider top-down approaches for choosing which states to split in order to generate improved policies (Munos and Moore, 1999). Recently (Khan et al., 2008) proposed an approach for the explanation of recommendations based on MDPs. They define a set of preferred scenarios that correspond to set of states with high expected utility, and generate explanations in terms of actions that will produce a preferred scenario based on predefined templates. They demonstrate their approach in the domain of course selection for students, modeled as a finite horizon MDP with three time steps. Thus, their is very limited previous work on explanation generation for decision-theoretic systems based on MDPs. In particular, there is no previous work on determining the relevant variable, which is the focus of this paper.

3 Factored 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 Φ that maps states **S** and actions **A** to new states **S'**. At each time, an agent receives a reward R that depends on the current state s and the applied action a. Thus, the main problem is to find a control strategy or *policy* π that maximizes the expected reward V over time.

For the discounted infinite-horizon case with any given discount factor γ, there is a policy π^* that is optimal regardless of the starting state and that satisfies the *Bellman* equation (Bellman, 1957):

$$V^{\pi}(s) = max_a\{R(s,a)+\gamma \sum_{s' \in \mathbf{S}} P(s'|s,a)V^{\pi}(s')\} \quad (1)$$

Two methods for solving this equation and finding an optimal policy for an MDP are: (a) dynamic programming and (b) linear programming (Puterman, 1994).

In a factored MDP, the set of states is described via a set of random variables $\mathbf{S} = \{X_1, ..., X_n\}$, 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, when the set of states $\mathbf{S} = Dom(X_i)$ is exponentially large, it results impractical to represent the transition model explicitly as matrices. Fortunately, the framework of dynamic Bayesian networks (DBN) (Dean and Kanasawa, 1989) gives us the tools to describe the transition model concisely. In these representations, the post-action nodes (at the time t+1) contain smaller matrices with the probabilities of their values given their parents' values under the effects of an action. For a more detailed description of factored MDPs see (Boutilier et al., 1999).

4 Relevant Variable Selection

As mentioned before, our strategy for automatic explanation generation based on MDPs considers as a first step to find the most relevant variable V_R for certain state s and action a. All the explanations we obtained from the experts are based on a variable which they consider the most important under the current situation (state) and according to the optimal policy. Examples of some of these explanations in the power plant domain are given later on the paper. We expect that something similar may happen in other domains, so discovering the relevant variable is an important first step for policy explanation based on MDPs.

Intuitively we can think that the relevant variable is the one with greater effect on the expected utility, given the current state and the optimal policy. So as an approximation to estimating the impact of each factor X_i in the utility, we estimate how much the utility, V, will change if we vary the value for each variable, compared to the utility of the current state. This is done by maintaining all the other variables, X_j, $j \neq i$, fixed. The process is repeated for all the variables, and the variable with the highest difference in value is selected as the relevant variable. An alternative criteria is to consider the action changes. That is, if the optimal action, a^*, in the current state, s, will change if a variable, X_i, has a different value. The variable that implies more changes will be in this case the relevant variable.

Thus, we propose two heuristic rules to determine the most relevant variable for an MDP, one rule based on utility and other rule based on policy. Next we describe in detail each rule.

4.1 Rule 1: Impact on utility

The policy of an MDP is guided by the utility function, so the impact of a variable in the utility is an important aspect regarding its relevance for certain state. The idea is to evaluate how much will the utility function will change if we vary the value of one of the variables for the current state, keeping the other variables fixed. We analyze this potential change in utility for all the variables, and the one with the highest difference will be considered the most relevant variable.

Let us assume that the process is in state s, then we measure the relevance of a variable X_i for the state s based on utility, denoted by $rel_s^V(X_i)$, as:

$$rel_s^V(X_i) = \max_{s' \in neigh_{X_i}(s)} V(s') - \min_{s' \in neigh_{X_i}(s)} V(s') \quad (2)$$

where $neigh_{X_i}(s)$ is the set of states that take the same the values as s for all other variables X_j, $j \neq i$; and a different value for the variable of interest, X_i. That is, the maximum change in utility when varying the value of X_i with respect to its value under the current state s. This expression is evaluated for all the variables, and the one with the highest value is considered the most relevant for state s, according to the value criteria:

$$X_R^V = argmax_i(rel_s^V(X_i)), \forall(i) \quad (3)$$

4.2 Rule 2: Impact on the optimal action

The second heuristic rule for determining the most relevant variable consists in exploring the optimal policy to detect changes in the optimal action for the state. That is, for each variable we verify if the optimal action will change if we vary its current value, keeping the other variables fixed. The variable that has more potential changes in policy will be considered more relevant.

Let us assume that the MDP is in state s, then we measure the relevance of a variable X_i for the state s according to its impact on policy, denoted by $rel_s^A(X_i)$, as:

$$rel_s^A(X_i) = \#s' : s' \in neigh_{X_i}(s) \wedge \pi^*(s) \neq \pi^*(s') \quad (4)$$

where $neigh_{X_i}(s)$ is the set of states that take the same values as s in all the variables except in variable X_i, $\pi^*(s)$ is the optimal action under the current state, s, and $\pi^*(s')$ is the action that will be taken in the other states such that

$s' \in neigh_{X_i}(s)$. In other words, this function measures how much the actions change when varying the value of X_i with respect to its value under the current state s. This expression is evaluated for all the variables, and the one with the highest value is considered the most relevant for state s, according to the policy criteria:

$$X_R^A = argmax_i(rel_s^A(X_i)), \forall(i) \qquad (5)$$

We evaluated and compared both rules in a real scenario for training power plant operators, as described in the next section.

5 Experimental Results

First we describe the intelligent assistant in which we tested our method for explanation generation, and then the experiments comparing the automatic relevant variable selection against a domain expert.

5.1 Intelligent assistant for operator training

We have developed an intelligent assistant for operator training (IAOT) (Figure 1).

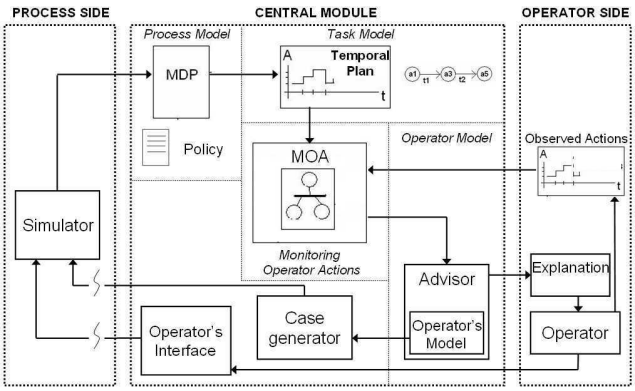


Figure 1: The intelligent assistant (IAOT) consists of 3 main parts: process side, operator side and central module. Based on the optimal policy obtained from the MDP a temporal plan is generated. The operator actions are compared to the plan and according to this the Adviser generates explanations

The input to the IAOT is a policy generated by a decision-theoretic planner (MDP), which establishes the sequence of actions that will allow to reach the optimal operation of a steam generator (Reyes et al., 2006). Operator actions are monitored and discrepancies are detected regarding the operator's expected behavior.

The process starts with an initial state of the plant, usually under an abnormal condition; so the operator should return the plant to its optimum operating condition using some controls. If the action performed by the operator deviates from the optimal plan, either in the type of action or its timing, an advice message is generated. Depending on the operator's performance, the adviser presents a new case through the case generator module.

We considered a training scenario based on a simulator of a combined cycle power plant, centered in the drum (a water tank) and the related control valves. Under certain conditions, the drum level becomes unstable and the operator has to return it to a safe state using the control valves. The variables in this domain are: (i) drum pressure (Pd), (ii) main steam flow (Fms), (iii) feed water flow (Ffw), (iv) generation (G), and (v) disturbance (this variable is not relevant for the explanations so is not included in the experiments). There are 5 possible actions: a0–do nothing, a1–increase feed water flow, a2–decrease feed water flow, a3–increase steam flow, and a4–decrease steam flow.

We started by defining a set of explanation units with the aid of a domain expert, to test their impact on operator training. These explanation units are stored in a data base, and the assistant selects the appropriate one to show to the user, according to the current state and optimal action given by the MDP. An example of an explanation unit is given in Figure 2. Each explanation unit has three main components: (i) the recommended action (upper left side), (ii) a verbal explanation of why this is the best action (lower left), and (iii) the relevant variable (V_R) highlighted in a schematic diagram of the process (right side). In this example the relevant variable is *generation*, $V_R = G$, as the absence of generation is the main reason to close the feed–water valve. Something similar occurs in all the explanation units.

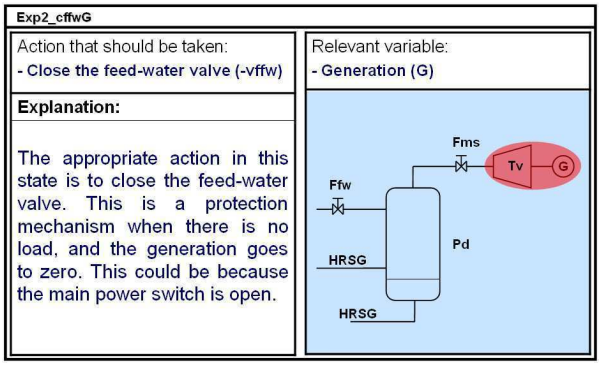


Figure 2: An example of an explanation unit.

To evaluate the effect of the explanations on learning, we performed a controlled experiment with 10 potential users with different levels of experience in power plant operation. The users were divided into two groups: G1, with explanations; G2, without explanations. Each participant has to control the plant to reach the optimal state under an emergency condition using a simulator and with the aid of the IAOT. During each session, the suggested actions and detected errors are given to the user, and for G1, also an explanation.

After some training sessions with the aid of the IAOT, the users were presented similar situations without the aid of the assistant. An analysis of the results (Elizalde et al., 2005) shows a significant difference in favor of the group with explanations. These results give evidence that explanations help in the learning of skills such as those required to operate an industrial plant.

As mentioned before, the explanations provided by the domain experts focus on the *most relevant variable*. In the next section we compare the relevant variables obtained by our method against those established by the human experts.

5.2 Results

Our main objective is to generate explanations that are similar to those given by a human expert, and in particular the identification of the most *relevant variable*. So to evaluate our methodology, we take as a reference the explanations given by the domain experts.

In the power plant domain there are 5 state variables (three binary variables, one with 6 values and other with 8 values), which makes a total of 384 states. We analyzed a random sample of 30 states, nearly 10% of the total number of states [1]. For the 30 cases we obtained the most relevant variable(s) based on both rules, according to their impact on utility and on policy; and compared these with the relevant variables given in the explanation units provided by the expert.

Figure 3 summarizes the results of the evaluation of the 30 cases. For each case we show: (i) the current state, (ii) the value, (iii) the optimal action, (iv) the variable selected according to the change in utility, including this change, (v) the number of changes in action for each variable (the highest are highlighted), and (vi) the relevant variable(s) given by the expert. Note that for some cases the expert gives two relevant variables.

From the table we observe that the rule based on the utility impact selects in 100% the most relevant variable according to the expert. The other rule, based on changes in policy, detects more than one variable in several cases. If we consider the subset of variables with highest number of changes in policy, at least one of the relevant variables given by the expert is contained in this subset for 80% of the cases. Both rules give a good match with the experts' selections, although the one based on utility is more specific and also more accurate. These are very promising results, as the method is giving, in general, the expected relevant variable, which is an important first step for producing automatic explanations based on MDPs.

[1] In our current implementation of the method its takes about half an hour to evaluate the impact on utility and policy per state, as we are transforming the MDP to an influence diagram and doing the calculations in Elvira (Elvira-Consortium, 2002); so it is not practical to consider all the states. In the future we plan to implement the method directly on the MDP to make it more efficient.

Test	selected S Var fms	ffw	d	pd	g	U	Π*	Experimental results Changes in Utility |ΔU|	Changes in actions # of changes to π* fms	ffw	pd	g	Relevant Var Selected by Expert
1	0	0	0	0	1	2601.29	a1	g = 2132.44	4/5	1/1	2/7	1/1	g,fms
2	0	0	1	3	1	2514.00	a2	g =2235.79	4/5	1/1	4/7	1/1	g,fms
3	1	1	0	4	0	796.95	a2	fms = 2413.53	2/5	0/1	1/7	1/1	fms, g
4	2	0	0	7	1	3488.60	a4	pd =2762.60	4/5	1/1	3/7	0/1	pd, fms
5	3	0	0	0	1	3295.55	a3	g =2427.94	5/5	1/1	2/7	1/1	g, pd
6	3	1	0	2	1	3053.19	a2	g = 2109.73	2/5	1/1	2/7	1/1	g, pd
7	4	0	1	0	1	2986.18	a1	g = 2257.11	4/5	1/1	2/7	1/1	g, pd
8	4	1	1	7	0	843.27	a4	pd = 3271.43	1/5	0/1	2/7	1/1	pd, g
9	5	1	0	1	1	3632.66	a0	fms = 3720.05	1/5	0/1	7/7	1/1	fms
10	5	1	1	1	1	3287.13	a0	fms = 3642.53	1/5	0/1	7/7	1/1	fms
11	0	0	0	4	0	468.85	a2	g = 2295.48	4/5	0/1	2/7	1/1	g
12	0	0	0	5	1	2624.39	a0	g = 2155.54	4/5	0/1	3/7	1/1	g, fms
13	0	0	0	6	1	2640.49	a0	g = 2171.64	5/5	1/1	3/7	1/1	g, fms
14	0	0	0	7	1	2601.29	a1	g = 2132.44	4/5	1/1	5/7	1/1	g, fms
15	0	1	0	5	0	468.85	a2	g = 2150.45	5/5	1/1	7/7	1/1	g, pd
16	0	1	1	7	0	566.76	a4	g = 2036	1/5	1/1	2/7	1/1	g
17	1	0	0	5	1	2486.17	a2	fms = 2116.43	5/5	0/1	5/7	0/1	fms
18	1	0	1	4	0	794.72	a4	pd = 3762.20	2/5	0/1	1/7	1/1	pd, g
19	1	1	0	5	0	766.95	a2	g = 1719.22	0/5	0/1	1/7	0/1	g
20	1	1	1	7	0	819.82	a4	pd = 3666.26	1/5	0/1	1/7	1/1	pd, g
21	2	0	0	1	1	6251.20	a0	g = 5394.74	1/5	0/1	5/7	1/1	g
22	2	1	0	2	1	3484.47	a2	pd = 2685.48	2/5	1/1	2/7	1/1	pd, ffw
23	2	1	0	6	0	850.83	a2	g = 2783.64	0/5	1/1	2/7	0/1	g
24	2	1	1	1	1	6180.16	a0	g = 5004.11	1/5	0/1	6/7	1/1	g, pd
25	3	0	0	4	1	3295.55	a3	g = 2427.94	4/5	1/1	2/7	1/1	g, pd
26	3	0	1	0	0	679.67	a3	pd = 2615.88	5/5	0/1	2/7	0/1	pd, g
27	3	1	1	2	1	3045.10	a2	pd = 1521.18	2/5	1/1	2/7	1/1	pd
28	4	0	0	0	1	2994.12	a1	g = 2135.25	4/5	1/1	2/7	1/1	g, pd
29	4	0	1	4	0	729.07	a4	pd = 3304.97	2/5	0/1	2/7	1/1	pd, g
30	5	0	0	5	1	2761.32	a4	fms = 2116.43	5/5	1/1	2/7	1/1	fms

Figure 3: This table summarizes the 30 cases in which we compared the relevant variable selected be each rule against those given by the expert.

6 Conclusions and Future Work

In this paper we have developed a method for determining the most relevant variable for generating explanations based on a factored MDP. The explanations provided by human experts are based on what they consider the most relevant variable in the current state, so obtaining this variable is an important first stage for automatic explanation generation. For determining the most relevant variable we proposed and compared two heuristic rules, one based on the impact on utility of each variable, and other based on their impact on the policy. We developed a method for finding the relevant variable based on these rules, and apply it to a realistic scenario for training power plant operators.

The experimental evaluation in the power plant domain shows that the methodology is promising, as the relevant variables selected agreed, in general, with those chosen by the expert. The rule based on utility impact seems more appropriate, at least in this domain, as it gives more specific results with a very high accuracy.

As future work we plan to integrate domain knowledge with the relevant variable obtained from the MDP to construct explanations; and test our method in other domains.

Acknowledgements

We thank the rest of the members of the Research Centre on Intelligent Decision-Support Systems (CISIAD) at UNED, in Madrid, Spain, and the power plant experts at the Instrumentation and Control Department of the Electrical Research Institute, Mexico. This project was supported in part by CONACYT under project No. 47968, and Francisco Elizalde by the Electrical Research Institute. The Spanish authors were supported by the Ministry of Education and Science (grant TIN-2006-11152). Manuel Luque was also partially supported by a predoctoral grant of the regional Government of Madrid.

References

R.E. Bellman. 1957. *Dynamic Programming*. Princeton U. Press, Princeton, N.J.

C. Bielza, J.A. Fernández del Pozo, and P. Lucas. 2003. Optimal decision explanation by extracting regularity patterns. In F. Coenen, A. Preece, and A.L. Macintosh, editors, *Research and Development in Intelligent Systems XX*, pages 283–294. Springer-Verlag.

C. Boutilier, T. Dean, and S. Hanks. 1999. Decision-theoretic planning: structural assumptions and computational leverage. *Journal of AI Research*, 11:1–94.

T. Dean and R. Givan. 1997. Model minimization in markov decision processes. In AAAI, editor, *In Proceedings AAAI-97*, pages 106–111, Cambridge, Massachusetts. MIT Press.

T. Dean and K. Kanasawa. 1989. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150.

M.J. Druzdzel. 1991. Explanation in probabilistic systems: Is it feasible? Will it work? In *Intelligent information systems V, Proceedings of the workshop*, pages 12–24, Poland.

F. Elizalde, E. Sucar, and P. deBuen. 2005. A prototype of an intelligent assistant for operator's training. In *International Colloquium for the Power Industry*, México. CIGRE-D2.

Elvira-Consortium. 2002. Elvira: An environment for creating and using probabilistic graphical models. In José A. Gómez and Antonio Salmerón, editors, *First European Workshop on Probabilistic Graphical Models, PGM'02*, pages 222–230, Cuenca, Spain.

R. Givan, T. Dean, and M. Greig. 2003. Equivalence notions and model minimization in markov decision processes. *Artif. Intell.*, 147(1-2):163–223.

J. Herrmann, M. Kloth, and F. Feldkamp. 1998. The role of explanation in an intelligent assistant system. In *Artificial Intelligence in Engineering*, volume 12, pages 107–126. Elsevier Science Limited.

O. Zia Khan, P. Poupart, and J. Black. 2008. Explaining recommendations generated by MDPs. In T. Roth-Berghofer et al., editor, *3rd International Workshop on Explanation-aware Computing ExaCt 2008*, Patras, Greece. Proceedings of the 3rd International ExaCt Workshop.

C. Lacave, R. Atienza, and F.J. Díez. 2000. Graphical explanations in Bayesian networks. In *Lecture Notes in Computer Science*, volume 1933, pages 122–129. Springer-Verlag.

C. Lacave, M. Luque, and F. J. Díez. 2007. Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics*, 37:952–965.

R. Munos and A.W. Moore. 1999. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1348–1355, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

M. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York.

S. Renooij and L. van der Gaag. 1998. Decision making in qualitative influence diagrams. In *Proceedings of the Eleventh International FLAIRS Conference*, pages 410–414, Menlo Park, California. AAAI Press.

A. Reyes, L. E. Sucar, E. Morales, and P. H. Ibarguengoytia. 2006. Solving hybrid Markov decision proceses. In *MICAI 2006: Advances in Artificial Intelligence*, Apizaco, Mexico. Springer-Verlag.

M. Wellman. 1990. Graphical inference in qualitative probabilistic networks. *Networks*, 20:687–701.

Learning naïve Bayes regression models with missing data using mixtures of truncated exponentials

Antonio Fernández
Department of Statistics & Applied Mathematics
University of Almería
04120 Almería, Spain
afalvarez@ual.es

Jens D. Nielsen
Computer Science Department
University of Castilla-La Mancha
02071 Albacete, Spain
dalgaard@dsi.uclm.es

Antonio Salmerón
Department of Statistics & Applied Mathematics
University of Almería
04120 Almería, Spain
antonio.salmeron@ual.es

Abstract

In the last years, mixtures of truncated exponentials (MTEs) have received much attention within the context of probabilistic graphical models, as they provide a framework for hybrid Bayesian networks which is compatible with standard inference algorithms and no restriction on the structure of the network is considered. Recently, MTEs have also been successfully applied to regression problems in which the underlying network structure is a naïve Bayes or a TAN. However, the algorithms described so far in the literature operate over complete databases. In this paper we propose an iterative algorithm for constructing naïve Bayes regression models from incomplete databases. It is based on a variation of the data augmentation method in which the missing values of the explanatory variables are filled by simulating from their posterior distributions, while the missing values of the response variable are generated from its conditional expectation given the explanatory variables. We illustrate through a set of experiments with various databases that the proposed algorithm behaves reasonably well.

1 Introduction

In the last years, mixtures of truncated exponentials (MTEs) (Moral et al., 2001) have received much attention within the context of probabilistic graphical models, as they provide a framework for hybrid Bayesian networks which is compatible with standard inference algorithms and no restriction on the structure of the network is imposed (Cobb and Shenoy, 2006; Rumí and Salmerón, 2007). Recently, MTEs have also been successfully applied to regression problems in which the underlying network structure is a naïve Bayes (Morales et al., 2007) or a tree augmented naïve Bayes (TAN) (Fernández et al., 2007). However, the algorithms described so far in the literature operate over complete databases. In this paper we propose an iterative algorithm for constructing naïve Bayes regression models from incomplete databases. It is based on a variation of the data augmentation method (Tanner and Wong, 1987) in which the missing values of the explanatory variables are filled by simulating from their posterior distributions, while the missing values of the response variable are generated from its conditional expectation given the explanatory variables.

The rest of the paper is organised as follows. The MTE model, which is the basis of our work, is described in Sec. 2. We analyse the background behind existing regression models using

MTEs in Sec. 3, and out of that analysis, we describe our new algorithm that operates over missing values. The behaviour of the algorithm is tested through two experiments in Sec. 4. The paper ends with the concluding remarks in Sec. 5.

2 The MTE model

We denote random variables by capital letters, and their values by lowercase letters. We use boldfaced characters to represent random vectors and their values. The support of the variable $\mathbf{X}$ is denoted by $\Omega_{\mathbf{X}}$. A potential of class MTE (Moral et al., 2001) is defined as follows:

Definition 1. (MTE potential) Let $\mathbf{X}$ be a mixed n-dimensional random vector. Let $\mathbf{W} = (W_1, \ldots, W_d)$ and $\mathbf{Z} = (Z_1, \ldots, Z_c)$ be the discrete and continuous parts of $\mathbf{X}$, respectively, with $c + d = n$. We say that a function $f : \Omega_{\mathbf{X}} \mapsto \mathbb{R}_0^+$ is a *Mixture of Truncated Exponentials potential (MTE potential)* if for each fixed value $\mathbf{w} \in \Omega_{\mathbf{W}}$ of the discrete variables $\mathbf{W}$, the potential over the continuous variables $\mathbf{Z}$ is defined as:

$$f(\mathbf{z}) = a_0 + \sum_{i=1}^{m} a_i \exp\left\{\sum_{j=1}^{c} b_i^{(j)} z_j\right\} \quad (1)$$

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. We also say that f is an MTE potential if there is a partition $D_1, \ldots, D_k$ of $\Omega_{\mathbf{Z}}$ into hypercubes and in each D_i, f is defined as in Eq. (1).

Definition 2. (MTE density) An MTE potential f is an *MTE density* if

$$\sum_{\mathbf{w} \in \Omega_{\mathbf{W}}} \int_{\Omega_{\mathbf{Z}}} f(\mathbf{w}, \mathbf{z}) d\mathbf{z} = 1 \ .$$

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 (Moral et al., 2001; Moral et al., 2003).

Example 1. Consider two continuous variables X and Y. A possible conditional MTE density for Y given X is the following:

$$f(y|x) = \begin{cases} 1.26 - 1.15e^{0.006y} \\ \quad \text{if } 0.4 \leq x < 5,\ 0 \leq y < 13 \ , \\ 1.18 - 1.16e^{0.0002y} \\ \quad \text{if } 0.4 \leq x < 5,\ 13 \leq y < 43 \ , \\ 0.07 - 0.03e^{-0.4y} + 0.0001e^{0.0004y} \\ \quad \text{if } 5 \leq x < 19,\ 0 \leq y < 5 \ , \\ -0.99 + 1.03e^{0.001y} \\ \quad \text{if } 5 \leq x < 19,\ 5 \leq y < 43 \ . \end{cases} \quad (2)$$

3 Regression using MTEs

Assume we have a set of variables $Y, X_1, \ldots, X_n$, where Y is continuous and the rest are either discrete or continuous. Regression analysis consists of finding a model g that explains the *response* variable Y in terms of the *explanatory* variables $X_1, \ldots, X_n$, so that given a configuration of the explanatory variables, $x_1, \ldots, x_n$, a prediction about Y can be obtained as $\hat{y} = g(x_1, \ldots, x_n)$. Previous work on regression using MTEs (Morales et al., 2007; Fernández et al., 2007) proceeds by representing the joint distribution of $Y, X_1, \ldots, X_n$ as a Bayesian network (naïve Bayes or TAN), and then using the posterior distribution of Y given $X_1, \ldots, X_n$ (more precisely, its expectation or its median) to obtain a prediction for Y.

3.1 Constructing a regression model from incomplete data

In this paper we will concentrate on the use of the expectation to analyse the regression problem with missing data. Therefore, our regression model will be

$$\hat{y} = g(x_1, \ldots, x_n) = E[Y|x_1, \ldots, x_n] = \int_{\Omega_Y} y f(y|x_1, \ldots, x_n) dy \ ,$$

where $f(y|x_1, \ldots, x_n)$ is the conditional density of Y given $x_1, \ldots, x_n$, which we assume to be of class MTE.

A conditional distribution of class MTE can be represented as in Eq. (2), where actually a marginal density is given for each element of the partition of the support of the variables involved. Within the context of regression, the distribution for the response variable Y given an element in a partition of the domain of the explanatory variables $X_1, \ldots, X_n$, can be regarded as an approximation of the true distribution of the actual values of Y for each possible configuration of the explanatory variables in that region of the partition. This fact justifies the selection of $E[Y|x_1, \ldots, x_n]$ as the predicted value for the regression problem, because that value is the one that best represents all the possible values of Y for that region, in the sense that it minimises the *mean squared error* between the actual value of Y and its predictions $\hat{y}$, namely

$$\text{mse} = \int_{\Omega_Y} (y - \hat{y})^2 f(y|x_1, \ldots, x_n) dy \ , \qquad (3)$$

which is known to be minimised for $\hat{y} = E[Y|x_1, \ldots, x_n]$. Therefore, the key point to find a regression model of this kind is to obtain a good estimation of the distribution of Y for each region of values of the explanatory variables. For the complete data case, the problem was studied in (Morales et al., 2007; Fernández et al., 2007), but the estimation of MTE distributions in the presence of missing data has not yet been addressed, but in the more restricted setting of unsupervised data clustering (Gámez et al., 2006). In that case, the only missing values are on the class variable, which is hidden, while the data about the features are complete.

Here we are interested in problems where the missing values can appear in the response variable as well as in the explanatory variables. A first approach to solve this problem could be to apply the EM algorithm (Dempster et al., 1977). However, the application of the methodology is problematic because the likelihood function for the MTE model cannot be optimised in an exact way (Rumí et al., 2006). Also, the aim of the EM algorithm is to find maximum likelihood estimates, which is not our main goal. From the point of view of regression, it is more important that the obtained models provide low values for the mean squared error rather than high likelihood.

Another way of approaching problems with missing values is the so-called *data augmentation* (DA) algorithm (Tanner and Wong, 1987). The advantage with respect to the EM algorithm is that DA does not require to directly optimise the likelihood function. Instead, it is based on imputing the missing values by simulating from the posterior distribution of the missing variables, which is iteratively improved from an initial estimation based on a random imputation. The DA algorithm leads to an approximation of the maximum likelihood estimates of the parameters of the model, as long as the parameters are estimated by maximum likelihood from the complete database in each iteration.

However, as we mentioned before, we are not so interested in the maximum likelihood estimates of the parameters of the model, but rather in reducing the mean squared error for the estimates of Y. With this aim, we show in the next proposition that using the conditional expectation of Y to impute the missing values instead of simulating values for Y (denoted as Y_S in the proposition), reduces the mse even if we simulate from the exact distribution of Y conditional on any configuration on a region of the values of the explanatory variables.

Proposition 1. *Let Y and Y_S be two continuous independent and identically distributed random variables. Then,*

$$E[(Y - Y_S)^2] \geq E[(Y - E[Y])^2] \ . \qquad (4)$$

Proof.

$$\begin{aligned} E[(Y - Y_S)^2] &= E[Y^2 + Y_S^2 - 2YY_S] \\ &= E[Y^2] + E[Y_S^2] - 2E[YY_S] \\ &= E[Y^2] + E[Y_S^2] - 2E[Y]E[Y_S] \\ &= 2E[Y^2] - 2E[Y]^2 \\ &= 2(E[Y^2] - E[Y]^2) = 2\text{Var}(Y) \\ &\geq \text{Var}(Y) = E[(Y - E[Y])^2] \ . \end{aligned}$$

□

In the proof we have used that both variables are independent and identically distributed, and therefore the expectation of the product is the product of the expectations, and the expected value of both variables is the same.

3.2 The algorithm for learning a NB regression model from incomplete data

Our proposal consists of an algorithm which iteratively learns a naïve Bayes regression model by imputing the missing values in each iteration according to the following criterion:

- If the missing value corresponds to the response variable, it is imputed with the conditional expectation of Y given the values of the explanatory variables in the same record of the database, computed from the current NB model.
- Otherwise, the missing cell is imputed by simulating the corresponding variable from its conditional distribution given the values of the other variables in the same record, computed from the current NB model.

As the imputation requires the existence of a model, more precisely a NB in our context, for the construction of the initial model we propose to impute the missing values by simulating from the marginal distribution of each variable computed from the observed values. In preliminary experiments we achieved better results using this alternative rather than pure random initialisation, which is the standard way of proceeding in data augmentation (Tanner and Wong, 1987). Another possibility is to simulate from the conditional distribution of each explanatory variable given the response, but the drawback is that the estimation of the conditional distributions requires more data than the estimation of the marginals, which can be problematic if the amount of missing values is high.

Therefore, the algorithm proceeds by imputing the initial database, learning an initial model and re-imputing the missing cells. Then, a new model is constructed and, if the mean squared error is reduced, the current model is

Algorithm 1: NB regression model from incomplete data

Input: An incomplete database D for variables $Y, X_1, \ldots, X_n$.
Output: A naïve Bayes regression model for response variable Y and explanatory variables $X_1, \ldots, X_n$.

for *each variable* $X \in \{Y, X_1, \ldots, X_n\}$ **do**
  Learn a univariate distribution $f_X(x)$ from its observed values in D.
end
Create a database D' from D by imputing the missing values for each $X \in \{Y, X_1, \ldots, X_n\}$ sampling from $f_X(x)$.
Create a database D_t from D by discarding the records where Y is missing.
Learn a NB regression model M' from D'.
Let $srmse'$ be the sample root mean squared error of M' computed using D_t according to Eq. (5).
$srmse \leftarrow \infty$.
while $srmse' < srmse$ **do**
  $M \leftarrow M'$.
  $srmse \leftarrow srmse'$.
  Create a new database D' from D filling the missing values as follows:
  for *each variable* $X \in \{X_1, \ldots, X_n\}$ **do**
    for *each record* $\mathbf{z}$ *in* D *with missing value for* X **do**
      Obtain $f_X(x|\mathbf{z})$ by probability propagation in model M.
      Impute the missing value for X by simulating from $f_X(x|\mathbf{z})$.
    end
  end
  for *each record* $\mathbf{z}$ *in* D *with missing value for* Y **do**
    Obtain $f_Y(x|\mathbf{z})$ by probability propagation in model M.
    Impute the missing value for Y with $E_{f_Y}[Y|\mathbf{z}]$.
  end
  Re-estimate model M' from D'.
  Let $srmse'$ be the sample root mean squared error of M' computed using D_t.
end
return M

replaced and the process repeated until convergence. As the mse in Eq. (3) requires the knowledge of the exact distribution of Y conditional on each configuration of the explanatory variables, we use as error measure the sample root mean squared error, computed as

$$\text{srmse} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \ , \qquad (5)$$

where m is the sample size, y_i is the true value of Y for record i and $\hat{y}_i$ is its corresponding prediction through the regression model.

The details are given in Alg. 1. Notice that, in steps 6 and 23 the naïve Bayes model is learnt from a complete database, and therefore the existing estimation methods for MTEs can be used (Rumí et al., 2006; Morales et al., 2007).

4 Experimental evaluation

In order to test the performance of the proposed method, we have carried out two experiments over four databases. One database (`mte50`) is synthetic, sampled from an MTE distribution, taken from (Morales et al., 2007). The other three databases are available in the UCI (Blake and Merz, 1998) and StatLib (StatLib, 1999) repositories. A description of the used databases can be found in Tab. 1.

Database	Size	# Cont.	# Disc.
`bodyfat`	251	15	0
`boston`	452	11	2
`cloud`	107	6	2
`mte50`	50	3	1

Table 1: A description of the databases used in the experiments, with their size and number of continuous and discrete variables.

The first experiment was oriented to test whether the model behaves reasonably, in the sense that the error is directly related to the percentage of missing values. With that aim, each database was divided at random into two parts, one for training with a 70% of the records, and one for test, with the remaining records. Then, we randomly inserted missing values in the training databases, ranging from a percentage of 10% to 50%. Previously, for each database, we repeated 100 times the same operation, obtaining the curves displayed in Fig. 1. The points correspond to the average srmse over the same test data by the 100 models learnt, and over each point there is a 95% confidence interval for the mean. The shape of the curves shows the expected behaviour, with the error increasing, in general, as the rate of missing grows.

The graphs in Fig. 2 show the log-likelihood corresponding to the learnt models as a function of the rate of missing values. Even though the goal of our algorithm is not to find highly likely models, the behaviour of the curves is still coherent. As in Fig. 1, the points indicate the average log-likelihood over the test database for the 100 runs of the experiment, and the intervals are 95% confidence intervals computed using the 100 measurements.

The second experiment was oriented to compare the proposed model with the M5' algorithm. The M5' algorithm (Wang and Witten, 1997) is an improved version of the model tree introduced by Quinlan (Quinlan, 1992). The model tree is basically a decision tree where the leaves contain a regression model rather than a single value, and the splitting criterion uses the variance of the values in the database corresponding to each node rather than the information gain. We chose the M5' algorithm because it was the state-of-the-art in graphical models for regression, before the introduction of MTEs for regression in (Morales et al., 2007). We have used the implementation of that method provided by Weka 3.4.11 (Witten and Frank, 2005). Regarding the implementation of the NB model, we have included it in the Elvira software (Elvira Consortium, 2002), which can be downloaded from `http://leo.ugr.es/elvira`.

In this experiment we have used 10-fold cross validation to estimate the srmse. The missing cells in the databases where selected before running the cross validation, therefore, in this case both the training and test databases contain missing cells in each iteration of the cross validation. We discarded from the test set the records for which the value of Y was missing. If the missing cells in the test set correspond to explanatory variables, algorithm M5' imputes

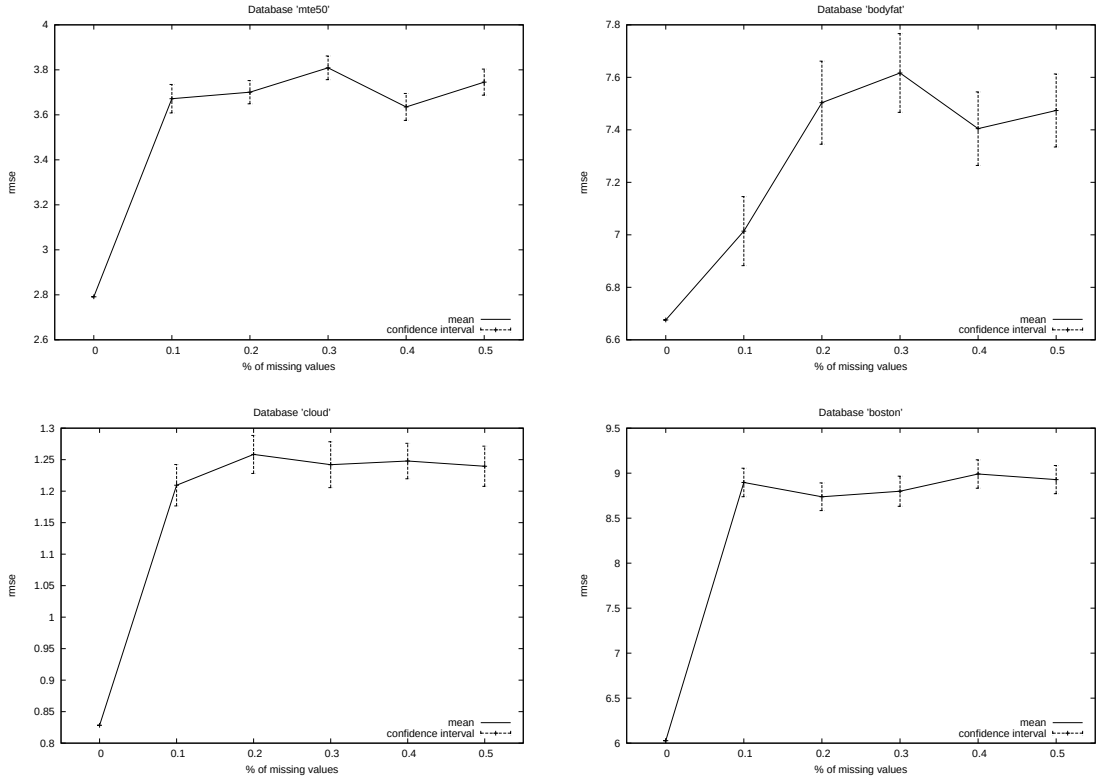


Figure 1: Sample root mean squared error as a function of the percentage of missing values.

them as the column average if the variable is continuous, or the column mode if it is qualitative (Witten and Frank, 2005). The NB model does not require the imputation of the missing explanatory variables in the test set, as the posterior distribution for Y is computed by probability propagation and therefore, the variables which are not observed are marginalised out. The results of the second experiment can be read in Tab. 2. The values displayed correspond to the average srmse computed by 10-fold cross validation. The result of the comparison is a draw, with NB winning for databases `bodyfat` and `mte50` and M5' winning in the other two cases. Friedman's test (Demsar, 2006) reports no statistically significant differences between both methods, with a p-value of 0.6831. This result was to be expected, as it is consistent with the comparison between models when they are learnt from complete datasets. It is surprising the error obtained by M5' for the database `bodyfat` with 50% of missing, which is much better than for lower rates of missing values. We think that this can be due to randomness.

4.1 Results discussion

The experiments carried out suggest that the proposed method behaves in a reasonable way. The graphs corresponding to the first experiment show a tendency of the error to increase along with the rate of missing values, except in some cases where it decreases around the 40% of missing, probably due to overfitting, as mentioned in (Friedman, 1997) for the general case of learning Bayesian networks. Also, we have similar graphs showing how the likelihood of the learnt models decrease as the rate of missing values increases.

Regarding the second experiment, the results are coherent with the ones obtained for the complete data case. However, we believe that our proposal should be superior to M5' in the case of learning from missing data. The reason is that we impute taking into account the condi-

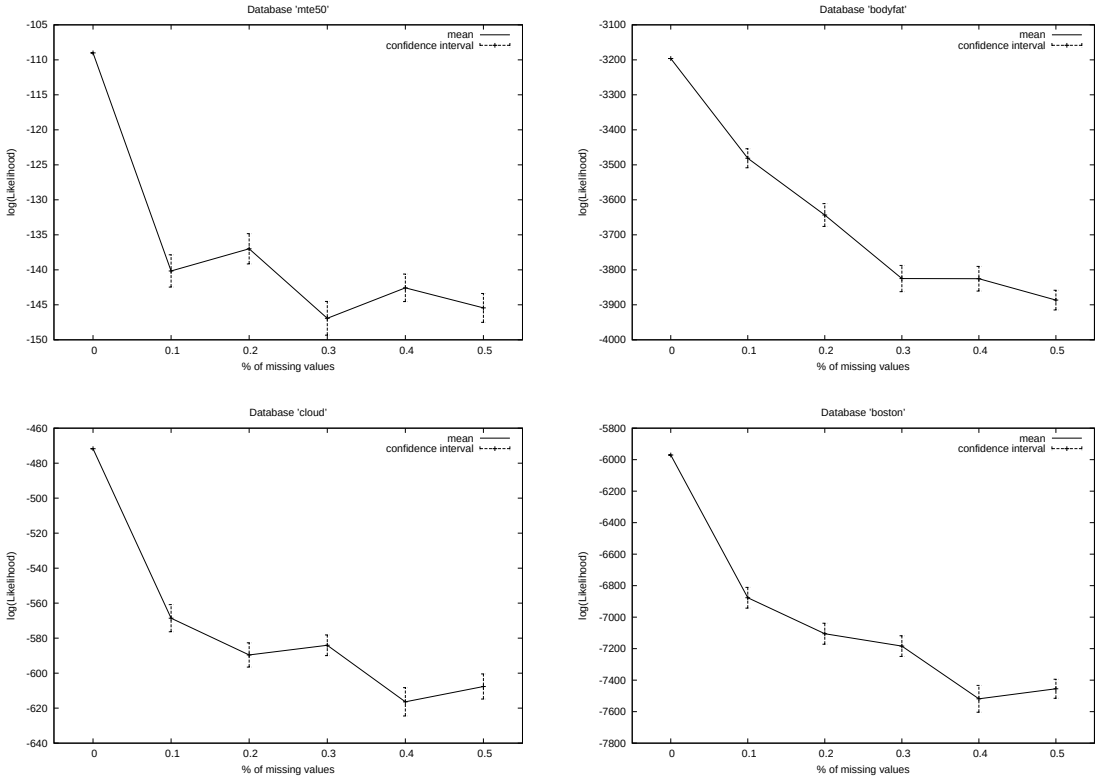


Figure 2: Loglikelihood of the learnt models as a function of the percentage of missing values.

tional distribution of the variable for which the missing value is going to be imputed, whilst M5' uses the marginal distribution. Our impression is that the reason why the results in these experiments are not even better for NB is the limited size of the used databases. Nevertheless, the independence assumptions in the NB model can be a limitation, and therefore more complex structures, like the TAN, might lead to more significant differences.

5 Concluding remarks

In this paper we have described a method for learning regression models from incomplete data based on the MTE distribution over a naïve Bayes network structure. The algorithm is supported by a result on how to minimise the prediction error and the experiments carried out, though somehow limited, show a reasonable performance of the new algorithm, compared to the robust M5' scheme, which is not surprising, as M5' is mainly designed for continuous explanatory variables. The behaviour of the algorithm is also good in terms of likelihood, even though that aspect is not really relevant to the aim of the method, which is to provide low prediction error.

The algorithm presented here can be improved in various ways, as for instance, by considering different manners of imputing the explanatory variables.

We think that the ideas contained in this work can be applied to other regression models like the TAN. However, the application to a broader problem like learning a Bayesian network of general purpose, is not straightforward, since in this case the goal would be to maximise a score based on the likelihood function, which requires maximum likelihood estimates of the parameters of the MTE model.

Acknowledgements

Work supported by the Spanish Ministry of Science and Innovation, projects TIN2007-67418-

		% of missing values					
Model	Database	0	0.1	0.2	0.3	0.4	0.5
NB	`bodyfat`	6.7095	6.3496	6.4602	6.6235	6.1287	6.9734
M5'	`bodyfat`	25.21	24.4519	29.0318	28.7724	28.6139	6.0929
NB	`boston`	6.2088	6.8668	6.4182	6.9748	7.0931	7.3654
M5'	`boston`	4.1475	5.1185	5.2011	5.6909	5.9646	6.6753
NB	`cloud`	0.5572	0.4897	0.6282	0.5350	0.7925	0.7137
M5'	`cloud`	0.3764	0.3237	0.6493	0.4421	0.4925	0.5919
NB	`mte50`	1.8695	2.0980	2.6392	2.7415	2.8957	3.0541
M5'	`mte50`	2.4718	2.7489	3.1566	2.6619	3.3681	3.4407

Table 2: Average srmse obtained in the experiment comparing NB vs. M5'.

C03-01 and TIN2007-67418-C03-02, and by Junta de Andalucía, project P05-TIC-00276.

References

C.L. Blake and C.J. Merz. 1998. UCI repository of machine learning databases. www.ics.uci.edu/∼mlearn/MLRepository.html. University of California, Irvine, Dept. of Information and Computer Sciences.

B. Cobb and P.P. Shenoy. 2006. Inference in hybrid Bayesian networks with mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 41:257–286.

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 B*, 39:1 – 38.

J. Demsar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1 – 30.

Elvira Consortium. 2002. Elvira: An environment for creating and using probabilistic graphical models. In J.A. Gámez and A. Salmerón, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models*, pages 222–230.

A. Fernández, M. Morales, and A. Salmerón. 2007. Tree augmented naive Bayes for regression using mixtures of truncated exponentials: Applications to higher education management. *IDA'07. Lecture Notes in Computer Science*, 4723:59–69.

Nir Friedman. 1997. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the ICML-97*.

J.A. Gámez, Rafael Rumí, and A. Salmerón. 2006. Unsupervised naive Bayes for data clustering with mixtures of truncated exponentials. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models (PGM'06)*, pages 123–132.

S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. *ECSQARU'01. Lecture Notes in Artificial Intelligence*, 2143:135–143.

S. Moral, R. Rumí, and A. Salmerón. 2003. Approximating conditional MTE distributions by means of mixed trees. *ECSQARU'03. Lecture Notes in Artificial Intelligence*, 2711:173–183.

M. Morales, C. Rodríguez, and A. Salmerón. 2007. Selective naive Bayes for regression using mixtures of truncated exponentials. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 15:697–716.

J.R. Quinlan. 1992. Learning with continuous classes. In *Procs. of the 5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore. World Scientific.

R. Rumí and A. Salmerón. 2007. Approximate probability propagation with mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 45:191–210.

R. Rumí, A. Salmerón, and S. Moral. 2006. Estimating mixtures of truncated exponentials in hybrid Bayesian networks. *Test*, 15:397–421.

StatLib. 1999. www.statlib.org. Department of Statistics. Carnegie Mellon University.

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.

Y. Wang and I.H. Witten. 1997. Induction of model trees for predicting continuous cases. In *Procs. of the Poster Papers of the European Conf. on Machine Learning*, pages 128–137.

I.H. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann.

The Probabilistic Interpretation of Model-based Diagnosis

Ildikó Flesch
MICC, Maastricht University,
Maastricht, the Netherlands
Email: ildiko@cs.ru.nl

Peter J.F. Lucas
Institute for Computing and Information Sciences,
Radboud University Nijmegen,
Nijmegen, the Netherlands
Email: peterl@cs.ru.nl

Abstract

Model-based diagnosis is the field of research concerned with the problem of finding faults in systems by reasoning with abstract models of the systems. Typically, such models offer a description of the structure of the system in terms of a collection of interacting components. For each of these components it is described how they are expected to behave when functioning normally or abnormally. The model can then be used to determine which combination of components is possibly faulty in the face of observations derived from the actual system. There have been various proposals in literature to incorporate uncertainty into the diagnostic reasoning process about the structure and behaviour of systems, since much of what goes on in a system cannot be observed. This paper proposes a method to decompose the probability distribution in probabilistic model-based diagnosis, partly in terms of the Poisson-binomial probability distribution.

1 Introduction

Almost from the inception of the field of probabilistic graphical models, Bayesian networks have been popular as formalisms to built *model-based*, diagnostic systems (Pearl, 1988). An alternative theory of model-based diagnosis was developed at approximately the same time, founded on techniques from logical reasoning (Reiter, 1987; de Kleer et al., 1992). The General Diagnostic Engine, GDE for short, is a well-known implementation of the logical theory; however, it also includes a restricted form of uncertainty reasoning to focus the diagnostic reasoning process (de Kleer and Williams, 1987). Previous research by Geffner and Pearl proved that the GDE approach to model-based diagnosis can be equally well dealt with by Bayesian networks (Geffner and Pearl, 1987; Pearl, 1988). Geffner and Pearl's results is basically a mapping from the logical representation as traditionally used within the logical diagnosis community to a specific Bayesian-network representation. The theory of model-based diagnosis supports multiple-fault diagnoses, which are similar to maximum a posteriori hypotheses, MAP hypotheses for short, in Bayesian networks (Gámez, 2004). Thus, although the logical and the probabilistic theory of model-based diagnosis have different origins, they are closely related.

Any theory of model-based diagnosis should consist of two parts: (*i*) a theory of how to *construct* models that can be used for diagnosing problems; (*ii*) a theory of how to *use* models to compute diagnoses. Whereas in both logic-based and probabilistic diagnosis issues concerning representation are clear—basically, logical expressions versus Bayesian networks—there is less clarity with regard to computing diagnoses. In logical model-based diagnosis there is a huge amount of literature investigating logical properties of diagnoses. In contrast, in literature on probabilistic diagnosis the emphasis is mostly on algorithmic properties of computing MAP diagnoses, rather then on probabilistic properties.

This paper proposes a new way to look at model-based diagnosis, taking the Bayesian-network representation by Geffner and Pearl as

the point of departure (Geffner and Pearl, 1987; Pearl, 1988). It is shown that by adding probabilistic information to a model of a system, the predictions that can be made by the model can be decomposed into a logical and a probabilistic part. The logical specifications are determined by the system components that are assumed to behave normally, constituting part of the system behaviour. This is complemented by uncertainty about behaviour for components that are assumed to behave abnormally. In addition, it is shown that the Poisson-binomial distribution plays a central role in determining model-based diagnoses. The results of this paper establish new links between traditional logic-based diagnosis, Bayesian networks and probability theory.

2 Poisson-binomial Distribution

First, we begin by summarising some of the relevant theory of discrete probability distributions (cf. (Cam, 1960; Darroch, 1964)).

Let $s = (s_1, \ldots, s_n)$ be a Boolean vector with elements $s_k \in \{0, 1\}$, $k = 1, \ldots, n$, where s_k is a Bernoulli discrete random variable and is equal to the outcome of trial k being either success (1) or failure (0). Let the probability of success of trial k be indicated by $p_k \in [0, 1]$ and the probability of failure be set to $1 - p_k$. Then, the probability of obtaining vector s as outcome is equal to

$$P(s) = \prod_{k=1}^{n} p_k^{s_k}(1 - p_k)^{1-s_k}. \quad (1)$$

This probability distribution acts as the basis for the *Poisson-binomial distribution*. The Poisson binomial distribution is employed to describe the outcomes of n independent Bernoulli distributed random variables, where only the number of success and failure are counted. The probability that there are m, $m \leq n$, successful outcomes amongst the n trials performed is then defined as:

$$f(m; n) = \sum_{s_1+\cdots+s_n=m} \prod_{k=1}^{n} p_k^{s_k}(1 - p_k)^{1-s_k}, \quad (2)$$

where f is a probability function. Here, the summation means that we sum over all the possible values of elements of the vector s, where the sum of the values of the elements must be equal to m.

It is easy to check that when all probabilities p_k are equal, i.e. $p_1 = \cdots = p_n = p$, where p denotes this identical probability, then the probability function $f(m; n)$ becomes that of the well-known *binomial distribution*:

$$g(m; n) = \binom{n}{m} p^m (1 - p)^{n-m}. \quad (3)$$

Finally, suppose that we model interactions between the outcomes of the trials by means of a Boolean function b. This means that we have an oracle that is able to observe the outcomes, and then gives a verdict whether the overall outcome is successful. The *expectation* or *mean* of this Boolean function is then equal to:

$$\mathcal{E}_P(b(S)) = \sum_{s} b(s)P(s). \quad (4)$$

with P defined according to Equation (1). This expectation also acts as the basis for the theory of causal independence, where a causal process is modelled in terms of interacting independent processes (cf. (Lucas, 2005)). Note that for $b(s) = b_m(s) \equiv s_1 + \cdots + s_n = m$ (i.e., the Boolean function that checks whether the number of successful trials is equal to m), we have that $\mathcal{E}_P(b_m(S)) = f(m; n)$. Thus, Equation (4) can be looked on as a generic way to combine the outcome of independent trials.

In the theory of model-based diagnosis, it is common to represent models of systems by means of logical specifications, which are equivalent to Boolean functions. Below, it will become clear that if we interpret the success probabilities p_k as the probability of observing the expected output of a system's component under the assumption that the component is faulty, then the theory of Poisson-binomial distributions can be used to describe part of probabilistic model-based diagnosis. However, first the necessary background to model-based diagnosis research is reviewed.

3 Uncertainty in Model-based Diagnosis

3.1 Model-based Diagnosis

In the theory of model-based diagnosis (Reiter, 1987), the structure and behaviour of a system is represented by a *logical diagnostic system* $\mathcal{S}_L = (\text{SD}, \text{COMPS})$, where

- SD denotes the *system description*, which is a finite set of logical formulae, specifying structure and behaviour;
- COMPS is a finite set of constants, corresponding to the *components* of the system; these components can be faulty.

The system description consists of *behaviour descriptions* and *connections*. A behavioural description is a formula specifying *normal* and *abnormal* (faulty) functionalities of the components. A connection is a formula of the form $i_c = o_{c'}$, where i_c and $o_{c'}$ denote the input and output of components c and c', respectively. This way an equivalence relation on the inputs and outputs is defined, denoted by $\text{IO}_{\setminus\equiv}$. The class representatives from this set are denoted by $[r]$.

A *logical diagnostic problem* is defined as a pair $\mathcal{P}_L = (\mathcal{S}_L, \text{OBS})$, where $\mathcal{S}_L$ is a logical diagnostic system and OBS is a finite set of logical formulae, representing *observations*.

Adopting the definition from (de Kleer et al., 1992), a diagnosis in the theory of consistency-based diagnosis is defined as follows. Let Δ be the assignment of either a normal or an abnormal behavioural assumption to *each* component. Then, Δ is a *consistency-based diagnosis* of the logical diagnostic problem $\mathcal{P}_L$ iff the observations are consistent with both the system description and the diagnosis:

$$\text{SD} \cup \Delta \cup \text{OBS} \nvDash \bot. \tag{5}$$

Here, $\nvDash$ stands for the negation of the logical entailment relation, and $\bot$ represents a contradiction.

EXAMPLE 1 Consider the logical circuit depicted in Figure 1, which represents a full adder,

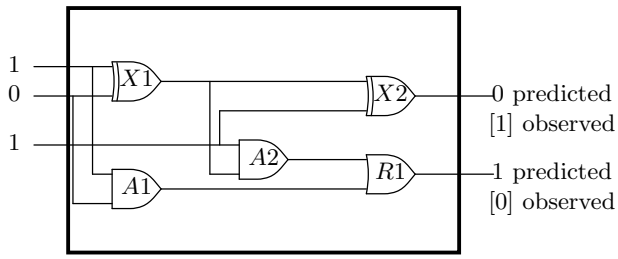


Figure 1: Full adder with inputs $\{i_1, \bar{i}_2, i_3\}$ and observed and predicted outputs.

i.e. a circuit that can be used for the addition of two bits with carry-in and carry-out bits. It is an example frequently used to illustrate concepts from model-based diagnosis. This circuit consists of two AND gates ($A1$ and $A2$), one OR gate ($R1$) and two exclusive-OR (XOR) gates ($X1$ and $X2$). These are the components that can be either faulty (abnormal) or normal. □

3.2 Probabilistic Model-based Diagnosis

In this section, we will map logical diagnostic problems onto probabilistic representations, called *Bayesian diagnostic problems*, using the Bayesian-network representation proposed by Geffner and Pearl (Geffner and Pearl, 1987; Pearl, 1988). As will become clear, a Bayesian diagnostic problem is defined as (i) a Bayesian diagnostic system representing the components, including their behaviour and interaction, based on information from the logical diagnostic system of concern, and (ii) a set of observations.

3.2.1 Graphical Representation

First the graphical structure used to represent the structural information from a logical diagnostic system is defined. It has the form of an acyclic directed graph $G = (V, E)$, where V is the set of *vertices* and $E \subseteq (V \times V)$ the set of directed edges, or *arcs* as we shall say.

Definition 1 ***(diagnostic mapping)*** *Let $\mathcal{S}_L = (\text{SD}, \text{COMPS})$ be a logical diagnostic system. The* diagnostic mapping *m_d maps $\mathcal{S}_L$ onto an acyclic directed graph $G = m_d(\mathcal{S}_L)$, as follows (see Figure 2):*

- *The vertices V of graph G are created according to the following rules:*

– *Each component* $c \in \text{COMPS}$ *yields a vertex* A_c *used to represent its normal and abnormal behaviour;*
– *Each class representative of an input or output* $[r] \in \text{IO}_{\setminus\equiv}$ *yields an associated vertex* $[r]$.

The set of all abnormality vertices A_c *is denoted by* Δ*, i.e.* $\Delta = \{A_c \mid c \in \text{COMPS}\}$. *The vertices of graph* G *are, thus, obtained as follows:*

$$V = \Delta \cup \text{IO}_{\setminus\equiv},$$

where $\text{IO}_{\setminus\equiv} = I \cup O$*, with disjoint sets* I *of* input vertices *and* O *of* output vertices.

- *The arcs* E *of* G *are constructed as follows:*
  – *There is an arc from each each input of a component* c *to each output of the component;*
  – *There is an arc for each component* c *from* $A_c \in V$ *to the corresponding output of the component.*

An example of this diagnostic mapping is shown in the following example.

EXAMPLE 2 Figure 3 shows the graphical representation of the full-adder circuit from Figure 1. The set V of vertices is:

$$\begin{aligned} V &= \Delta \cup O \cup I \\ &= \{A_{X1}, A_{X2}, A_{A1}, A_{A2}, A_{R1}\} \\ &\quad \cup \{O_{X1}, O_{X2}, O_{A1}, O_{A2}, O_{R1}\} \\ &\quad \cup \{I_1, I_2, I_3\}. \end{aligned}$$

The arcs from E connect (i) outputs of two components such as $O_{X1} \rightarrow O_{X2}$, (ii) an abnormality vertex with an output vertex such as $A_{A2} \rightarrow O_{A2}$ and (iii) an input vertex with an output vertex such as $I_3 \rightarrow O_{X2}$. □

3.2.2 Bayesian Diagnostic Problems

Recall that Bayesian networks that act as the basis for diagnostic Bayesian networks consist of two parts: a joint probability distribution and a graphical representation of the relations among the random variables defined by the joint probability distribution. Based on the definition of

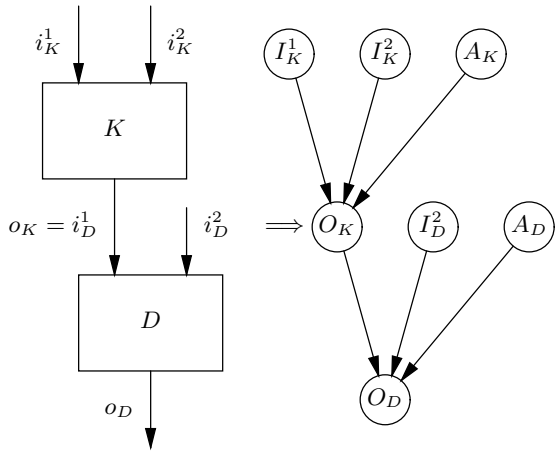


Figure 2: The diagnostic mapping.

Bayesian networks, particular parts of a logical diagnostic system will be related to the graphical structure of a diagnostic Bayesian network, whereas other parts will have a bearing on the content of the probability table of the Bayesian network.

Having introduced the mapping of a logical diagnostic system to its associated graph structure, we next introduce the full concept of a Bayesian diagnostic system.

Definition 2 ***(Bayesian diagnostic system)*** *Let* $\mathcal{S}_L = (\text{SD}, \text{COMPS})$ *be a logical diagnostic system, and* $G = m_d(\mathcal{S}_L)$ *be obtained by applying the diagnostic mapping. Let* P *be a joint probability distribution of the vertices of* G*, interpreted as random variables. Then,* $\mathcal{S}_B = (G, P)$ *is the associated* Bayesian diagnostic system.

Recall that by the definition of a Bayesian network, the joint probability distribution P of a Bayesian diagnostic system can be factorised as follows:

$$P(I, O, \Delta) = \prod_c P(O_c \mid \pi(O_c)) P(I) P(\Delta), \quad (6)$$

where O_c is an output variable associated to component $c \in \text{COMPS}$, and $\pi(O_c)$ are the random variables corresponding to the *parents* of O_c. The parents will normally consist of inputs I_c and an abnormality variable A_c.

To simplify notation, in the following, (sets of) random variables of a Bayesian diagnos-

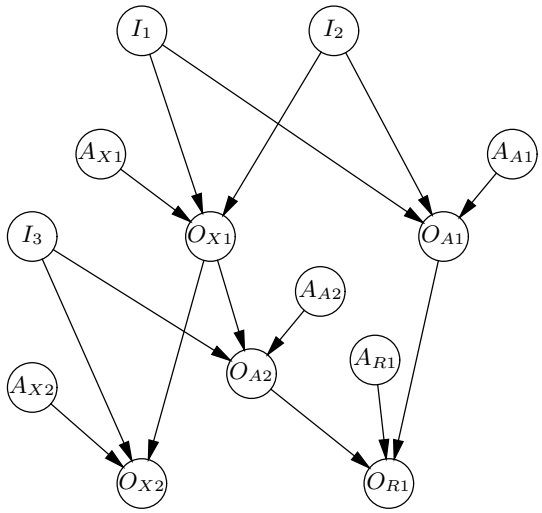


Figure 3: A Bayesian diagnostic system corresponding to the circuit in Figure 1.

tic problem have the same names as the corresponding vertices. By a_c is indicated that abnormality variable A_c takes the value 'true', whereas by $\bar{a}_c$ it is indicated that A_c takes the value 'false'. A similar notation will be used for the other random variables. Finally, a specific abnormality assumption concerning all abnormality variables is denoted by δ_C, which is defined as follows:

$$\delta_C = \{a_c \mid c \in C\} \cup \{\bar{a}_c \mid c \in \text{COMPS} - C\},$$

with $C \subseteq$ COMPS. There are some sensible constraints on the joint probability distribution P of a Bayesian diagnosis system that can be derived from the specification of the logical diagnostic system. These will be discussed later.

As with logical diagnostic problems, we need to add observations to Bayesian diagnostic systems in order to be able to solve diagnostic problems. In logical diagnostic systems, observations are the inputs and outputs of a system. It is generally not the case that the entire set of inputs and outputs of a system is observed. The set of input and output variables that have been observed, are referred to by I_ω and O_ω, respectively. The unobserved input and output variables will be referred to as I_u and O_u, respectively. We will use the notation i_ω to denote the values of the observed inputs, and o_ω for the observed output values. The set of *observations* is then denoted as $\omega = i_\omega \cup o_\omega$.

Now, we are ready to define the notion of Bayesian diagnostic problem, which is a Bayesian diagnostic system augmented by a set of observations.

Definition 3 ***(Bayesian diagnostic problem)*** *A* Bayesian diagnostic problem*, denoted by $\mathcal{P}_B$, is defined as the pair $\mathcal{P}_B = (\mathcal{S}_B, \omega)$, where $\mathcal{S}_B$ is a Bayesian diagnostic system and ω the set of observations of this system.*

Determining the diagnoses of a Bayesian diagnostic problem amounts to computing $P(\delta_C \mid \omega)$, and then finding the δ_C which maximises $P(\delta_C \mid \omega)$, i.e.

$$\delta_C^* = \arg\max_{\delta_C} P(\delta_C \mid \omega).$$

This problem is NP-hard (Gámez, 2004). The probability $P(\delta_C \mid \omega)$ can be computed by Bayes' rule, using the probabilities from the specification of a Bayesian diagnostic system:

$$P(\delta_C \mid \omega) = \frac{P(\omega \mid \delta_C)P(\delta_C)}{P(\omega)}. \tag{7}$$

As a consequence of the independences that hold for a Bayesian diagnostic system, it is possible to simplify the computation of the conditional probability distribution $P(\omega \mid \delta_C)$. According to the definition of a Bayesian diagnostic system it holds that

$$P(i \mid \delta_C) = P(i) \ ,$$

for each $i \subseteq (i_\omega \cup i_u)$, as the input variables and abnormality variables are independent. In addition, it is assumed that the input variables are independent.

Using these results, basic probability theory and the definition of a Bayesian diagnostic problem yields the following derivation:

$$\begin{aligned} P(\omega \mid \delta_C) &= P(i_\omega, o_\omega \mid \delta_C) \\ &= \sum_{i_u} P(i_u) P(i_\omega, o_\omega \mid i_u, \delta_C) \\ &= P(i_\omega) \sum_{i_u} P(i_u) \\ &\quad \times \sum_{o_u} \prod_{c} P(O_c \mid \pi(O_c)), \end{aligned} \tag{8}$$

since it holds by the axioms of probability theory that

$$P(i_\omega, o_\omega \mid i_u, \delta_C) = \sum_{o_u} P(i_\omega) \prod_c P(O_c \mid \pi(O_c)) \,.$$

To emphasise that the set of parents $\pi(O_c)$ includes an abnormality variable that is assumed to be true, i.e. the component is assumed to behave abnormally, the following notation is used $P(O_c \mid \pi(O_c) : a_c)$; similar, for the situation where the component c is assumed to behave normally the notation $P(O_c \mid \pi(O_c) : \bar{a}_c)$ is employed. Finally, the following assumptions, which will be used in the remainder of this paper, are made:

- $P(O_c \mid \pi(O_c) : a_c) = P(O_c \mid a_c)$, i.e. the probabilistic behaviour of a component that is faulty is independent of its inputs;
- $P(O_c \mid \pi(O_c) : \bar{a}_c) \in \{0, 1\}$, i.e. normal components behave deterministically.

The probability $P(o_c \mid a_c)$ will be abbreviated in the following section as p_c; thus $P(\bar{o}_c \mid a_c) = 1 - p_c$ These are realistic assumptions, as it is unlikely that detailed functional behaviour will be known for a component that is faulty, whereas when the component is not faulty, it is certain it will behave as intended.

4 Decomposition of Probability Distribution

To establish that probabilistic model-based diagnosis can be partly interpreted in terms of a Poisson-binomial distribution, it is necessary to decompose Equation (8) into various parts. The first part will represent the probabilities that components c produce the right, o_c, or wrong, $\bar{o}_c$, output, which correspond to the success and failure probabilities, respectively, of a Poisson-binomial distribution. The second part represents a normally functioning system fragment, which will be represented by a Boolean function. There is also a third part, which concerns the observed and unobserved inputs. We start by distinguishing between various types of components, inputs and outputs, in order to make the necessary distinction:

- The sets of components assumed to function *normally* and *abnormally* will be denoted by $C^{\bar{a}}$ and C^a, respectively, with $C^{\bar{a}}, C^a \subseteq \text{COMPS}$;
- The sets $C^{\bar{a}}$ and C^a are partitioned into sets of components, for *observed* and *unobserved* outputs, indicated by the sets $C^{\bar{a}}_\omega$, $C^{\bar{a}}_u$, C^a_ω and C^a_u, respectively.

Thus, $C^{\bar{a}} = C^{\bar{a}}_\omega \cup C^{\bar{a}}_u$ and $C^a = C^a_\omega \cup C^a_u$. In addition, we will sometimes make a distinction between components c for which o_c has been observed, and components c for which $\bar{o}_c$ has been observed. These sets will be denoted by C^o_ω and $C^{\bar{o}}_\omega$, respectively. It holds that C^o_ω and $C^{\bar{o}}_\omega$ constitute a partition of C_ω. The notations can also be combined, e.g., as $C^{a,o}_\omega$ and $C^{a,\bar{o}}_\omega$. Furthermore, we will sometimes use a similar notation for sets of output variables, e.g., $O^{\bar{a}}_u = \{O_c \mid c \in C^{\bar{a}}_u\}$ and $O^{\bar{a}}_\omega = \{O_c \mid c \in C^{\bar{a}}_\omega\}$, and input variables, e.g., $I^{\bar{a}}_u = \bigcup_{c \in C^{\bar{a}}_u} I_c$ indicates unobserved inputs of components that are assumed to behave normally and $I^{\bar{a}}_\omega = \bigcup_{c \in C^{\bar{a}}_\omega} I_c$ are observed inputs of components that are assumed to behave normally, with I_c the set of input variables of component $c \in \text{COMPS}$ and $I^{\bar{a}} = I^{\bar{a}}_\omega \cup I^{\bar{a}}_u$.

The following lemma shows that it is possible to decompose part of the joint probability distribution of Equation (6) using the component sets defined above.

Lemma 1 *The following statements hold:*

- *The joint probability distribution of the outputs of the set of assumed* normally *functioning components $C^{\bar{a}}$, can be decomposed into two products as follows:*

$$\prod_{c \in C^{\bar{a}}} P(O_c \mid \pi(O_c) : \bar{a}_c) = \prod_{c \in C^{\bar{a}}_u} P(O_c \mid \pi(O_c) : \bar{a}_c) \times \prod_{c \in C^{\bar{a}}_\omega} P(O_c \mid \pi(O_c) : \bar{a}_c).$$

- *Similarly, the joint probability distribution of the outputs of the set of assumed* abnormally *functioning components C^a, can be*

decomposed into two products as follows:

$$\prod_{c\in C^a} P(O_c \mid \pi(O_c) : a_c) = \prod_{c\in C^a_u} P(O_c \mid a_c) \prod_{c\in C^a_\omega} P(O_c \mid a_c).$$

Proof: The decompositions follows from the definitions of the sets C^a, C^a_ω, C^a_u, $C^{\bar{a}}_u$ and $C^{\bar{a}}_\omega$, and the independence assumptions underlying the distribution P. □

Now, based on Lemma 1, we can also decompose the product of the *entire* set of components, as follows:

$$\begin{aligned}&\prod_c P(O_c \mid \pi(O_c))\\ &= \prod_{c\in C^{\bar{a}}_u} P(O_c \mid \pi(O_c) : \bar{a}_c) \prod_{c\in C^{\bar{a}}_\omega} P(O_c \mid \pi(O_c) : \bar{a}_c)\\ &\quad \times \prod_{c\in C^a_u} P(O_c \mid a_c) \prod_{c\in C^a_\omega} P(O_c \mid a_c).\end{aligned}$$

Next, we show that the outputs of the set of observed abnormal components C^a_ω only depend on probabilities $p_c = P(o_c \mid a_c)$, $c \in C^a_\omega$.

Lemma 2 *The joint probability of observed outputs of the abnormally assumed components can be written as:*

$$\prod_{c\in C^a_\omega} P(O_c \mid \pi(O_c) : a_c) = \prod_{c\in C^{a,o}_\omega} p_c \prod_{c\in C^{a,\bar{o}}_\omega} (1 - p_c).$$

Proof: This follows straight from the definitions of C^a_ω, $C^{a,o}_\omega$ and $C^{a,\bar{o}}_\omega$. □

Recall that the probability of an output of a normally functioning component was assumed to be either 0 or 1, i.e. $P(O_c \mid \pi(O_c) : \bar{a}_c) \in \{0, 1\}$. Clearly, these probabilities yield, when multiplied, Boolean functions. One of these Boolean functions, denoted by φ, is defined as follows: $\varphi(o^{\bar{a}}_u, o^a_u, i^{\bar{a}}) = \prod_{c\in C^{\bar{a}}_u} P(O_c \mid \pi(O_c) : \bar{a}_c)$, where the set of parents $\pi(O_c)$ may, but need not, contain variables from the sets of variables O^a_u and $I^{\bar{a}}$. However, $\pi(O_c)$ does not contain variables from the set I^a, as these only condition variables that are assumed to behave abnormally and are then ignored, as mentioned at the end of the previous section. Similarly, we define Boolean functions $\psi(o_u, o^{\bar{a}}_\omega, i^{\bar{a}}) = \prod_{c\in C^{\bar{a}}_\omega} P(O_c \mid \pi(O_c) : \bar{a}_c)$.

Lemma 3 *For each value o^a_u and $i^{\bar{a}}$, there exists exactly* one *value $o^{\bar{a}}_u$ of the set of variables $O^{\bar{a}}_u = \{O_c \mid c \in C^{\bar{a}}_u\}$ for which it holds that $\varphi(o^a_u, o^{\bar{a}}_u, i^{\bar{a}}) = 1$; similarly, for each value o_u and $i^{\bar{a}}$ there exists one value $o^{\bar{a}}_\omega$ of the set of variables $O^{\bar{a}}_\omega = \{O_c \mid c \in C^{\bar{a}}_\omega\}$ for which it holds that $\psi(o_u, o^{\bar{a}}_\omega, i^{\bar{a}}) = 1$.*

Proof: As both the functions φ and ψ are defined as products of conditional probability distributions $P(O_c \mid \pi(O_c) : \bar{a}_c)$, for which we have that $P(o_c \mid \pi(O_c) : \bar{a}_c) \in \{0, 1\}$, there is, due to the axioms of probability theory, for any value of the variables corresponding to the parents of the variables O_c at most one value for each O_c for which the joint probability $\prod_c P(O_c \mid \pi(O_c) : \bar{a}_c) = 1$. □

The following lemma, which is used later, is a consequence of the definition of these Boolean functions.

Lemma 4 *Let the Boolean functions φ and ψ be as defined above, then:*

$$\sum_{o_u} \varphi(o^a_u, o^{\bar{a}}_u, i^{\bar{a}})\psi(o_u, o^{\bar{a}}_\omega, i^{\bar{a}}) \prod_{c\in C^a} P(O_c \mid a_c) = \sum_{o^a_u} b(o^a_u, i^{\bar{a}}) \prod_{c\in C^{a,o}} p_c \prod_{c\in C^{a,\bar{o}}} (1 - p_c),$$

with Boolean function b and $p_c = P(o_c \mid a_c)$.

Proof: For a fixed set of observed outputs o_ω let $b(o_u, i^{\bar{a}}) = \varphi(o^a_u, o^{\bar{a}}_u, i^{\bar{a}})\psi(o_u, o^{\bar{a}}_\omega, i^{\bar{a}})$, then,

$$\sum_{o_u} \varphi(o^a_u, o^{\bar{a}}_u, i^{\bar{a}})\psi(o_u, o^{\bar{a}}_\omega, i^{\bar{a}}) \prod_{c\in C^a} P(O_c \mid a_c) = \sum_{o_u} b(o_u, i^{\bar{a}}) \prod_{c\in C^a} P(O_c \mid a_c).$$

Furthermore, due to Lemma 3, it suffices to only consider the restriction of the function b to the variables O^a_u and $I^{\bar{a}}$, as for given values o^a_u and $i^{\bar{a}}$, $b(o^a_u, o^{\bar{a}}_u, i^{\bar{a}}) = 0$ for all but one value of $O^{\bar{a}}_u$. This function is denoted by $b(o^a_u, i^{\bar{a}})$. The product term results from application of a slight generalisation of Lemma 2. □

We are now ready to establish that $P(\omega \mid \delta_C)$ can be written as the sum of weighted products of the form $\prod_c p_c \prod_{c'}(1 - p_{c'})$.

Theorem 1 *Let $\mathcal{P}_B = (\mathcal{S}_B, \omega)$ be a Bayesian diagnostic problem. Then, $P(\omega \mid \delta_C)$ can be expressed as follows:*

$$P(\omega \mid \delta_C) = P(i_\omega) \sum_{i_u^{\bar{a}}} P(i_u^{\bar{a}}) \sum_{o_u^a} b(o_u^a, i^{\bar{a}}) \times \prod_{c \in C^{a,o}} p_c \prod_{c \in C^{a,\bar{o}}} (1 - p_c)$$

where $b(o_u^a, i^{\bar{a}}) \in \{0, 1\}$ and $p_c = P(o_c \mid a_c)$.

Proof: The result follows from the above lemmas and the fact that we sum over (part of) the input variables I. Note that only the variables $I^{\bar{a}}$ are used as conditioning variables, which follows from the assumption that $P(O_c \mid \pi(O_c) : a_c) = P(O_c \mid a_c)$. As only the input variables $i_u^{\bar{a}}$ are assumed to be dependent of output variables, we obtain: $\sum_{i_u, o_u^a} P(i_u) \cdots = \sum_{i_u^{\bar{a}}, o_u^a} P(i_u^{\bar{a}}) \cdots$. The Boolean function $b(o_u^a, i^{\bar{a}})$ is as above. □

An alternative version of the theorem can be obtained in terms of expectations using Equation (4) for the Poisson-binomial distribution:

$$P(i_\omega) \sum_{i_u^{\bar{a}}} P(i_u^{\bar{a}}) \sum_{o_u^a} b(o_u^a, i^{\bar{a}}) \prod_{c \in C^{a,o}} p_c \prod_{c \in C^{a,\bar{o}}} (1 - p_c)$$
$$= P(i_\omega) \prod_{c \in C_\omega^a} P(O_c \mid a_c) \sum_{i_u^{\bar{a}}} P(i_u^{\bar{a}}) \mathcal{E}_P(b_{i^{\bar{a}}}(O_u^a)),$$

i.e. the sum of the mean of the Boolean functions $b_{i^{\bar{a}}}$, which are functions of the unobserved inputs $i_u^{\bar{a}}$, in terms of the probability function P (Equation (4)), weighed by the prior probability of unobserved inputs $i_u^{\bar{a}}$. Combining this with Equation (7) yields $P(\delta_C \mid \omega)$. Thus, to probabilistically rank diagnoses δ_C it is necessary to compute: (i) $\mathcal{E}_P(b_{i^{\bar{a}}}(O_u^a))$, the Poisson-binomial distribution mean of the behaviour of the normally assumed components, (ii) $P(i_u^{\bar{a}})$, (iii) $\prod_{c \in C_\omega^a} P(O_c \mid a_c)$, the observed abnormal components, and (iv) the prior $P(\delta_c)$. Note that both $P(i_\omega)$ and $P(\omega)$ can be ignored.

5 Conclusions

We have shown that probabilistic model-based diagnosis, which is an extension of traditional GDE-like model-based diagnosis, can be decomposed into computation of various probabilities, in which a central role is played by the Poisson-binomial distribution. When all probabilities $p_c = P(o_c \mid a_c)$ are assumed to be equal, a common simplifying assumption in model-based diagnosis, the analysis reduces to the use of the standard binomial distribution.

So far, most other research on integrating probabilistic reasoning with logic-based model-based diagnosis took probabilistic reasoning as adding some sort of uncertain, abductive reasoning to logical reasoning. No attempts were made in related research to look inside what happens in the diagnostic process, as was done in this paper. We expect that it becomes thus possible to investigate further variations in probabilistic model-based diagnosis, for example, by adopting assumptions different from those in this paper with regard to fault behaviour in systems.

References

L. Le Cam. 1960. An approximation theorem for the poisson binomial distribution. *Pacific Journal of Mathematics*, 10:1181–1197.

J. Darroch. 1964. On the distribution of the number of successes in independent trials. *The Annals of Mathematical Statistics*, 35:1317–1321.

J. de Kleer and B. C. Williams. 1987. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130.

J. de Kleer, A. K. Mackworth, and R. Reiter. 1992. Characterizing diagnoses and systems. *Artificial Intelligence*, 52:197–222.

J.A. Gámez, 2004. *Abductive inference in Bayesian Networks: a review*, pages 101–120. Springer, Berlin.

H. Geffner and J. Pearl. 1987. Distributed diagnosis of systems with multiple faults. In *Proc. of the 3rd IEEE Conference on AI Applications*, pages 156–162. IEEE.

P.J.F. Lucas. 2005. Bayesian network modelling through qualitative patterns. *Artificial Intelligence*, 163:233–263.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffman, San Francisco, CA.

R. Reiter. 1987. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95.

Efficient Bayesian Network Learning Using EM or Pairwise Deletion

Olivier C. H. François

University of Reading, Sustainable Urban Environments Research Division,
URS building, 3n38, Whiteknights, PO Box 219, RG6 6AW, Reading, United Kingdom.
http://ofrancois.tuxfamily.org

Abstract

In previous work, we have seen how to learn a TAN classifier from incomplete dataset using the Expectation Maximisation algorithm (François and Leray, 2006). In this paper, we study differences for Bayesian network structure learning between estimating probabilities using the EM algorithm or using Pairwise Deletion. We have implemented these two estimation techniques with greedy search learning methods in several spaces: Trees, Directed Acyclic Graphs, Completed Partially Directed Acyclic Graphs or Tree Augmented Naive Bayes structures. An experimental study shows strengths and weaknesses of using the EM algorithm or Pairwise Deletion on classification tasks.

1 Introduction

Bayesian networks introduced by (Kim and Pearl, 1987) are a formalism of probabilistic reasoning used increasingly in decision aid, diagnosis and complex systems control (Jensen, 1996; Pearl, 1998; Naïm et al., 2004).

Let $\mathbb{X} = \{X_1, \ldots, X_n\}$ be a set of discrete random variables. A *Discrete Bayesian network* $\mathcal{B} =< \mathcal{G}, \Theta >$ is defined by a directed acyclic graph (DAG) $\mathcal{G} =< \mathbb{N}, \mathbb{U} >$ 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 $Pa(X_i)$ (with r_i and q_i as respective cardinalities of X_i and $Pa(X_i)$).

Determination of Θ (when $\mathcal{G}$ is given) is often based on expert knowledge, but several learning methods based on data have appeared. However, most of these methods only deal with complete data cases.

We will therefore first recall the issues relating to structural learning and review the various ways of dealing with incomplete data for structure determination.

Because of the super-exponential size of the search space (Gillispie and Lemieux, 2001), exhaustive search for the best Bayesian network 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 are -most of the time- completely observed.

We are here more specifically interested in score-based methods, primarily MWST proposed by (Chow and Liu, 1968) and applied to Bayesian networks in (Heckerman et al., 1995), then GS algorithm (Chickering et al., 1995), and finally GES algorithm proposed by (Chickering and Meek, 2002). GS is a greedy search carried out in DAG spaces where the interest of each structure located near the current structure is assessed by means of a Bayesian score like BDe (Heckerman et al., 1994) or a BIC/MDL type measurement (equation 1). As (Friedman, 1997), we consider that the BIC/MDL score is a function of the graph $\mathcal{G}$ and the parameters Θ, generalising the classical definition of the BIC score which is defined with our notation by $BIC(\mathcal{G}, \Theta^*)$ where Θ^* is obtained by maximising the likelihood or $BIC(\mathcal{G}, \Theta)$ score for a given $\mathcal{G}$ which is given by

$$BIC(\mathcal{G}, \Theta) = \log P(\mathcal{D}|\mathcal{G}, \Theta) - \frac{\log N}{2} \mathrm{Dim}(\mathcal{G}) \quad (1)$$

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 local score computed for each node of the graph:

$$BIC(\mathcal{G}, \Theta) = \sum_i bic(X_i, P_i, \Theta_{Xi|P_i}) \qquad (2)$$

where $bic(X_i, P_i, \Theta_{Xi|P_i}) =$

$$\sum_{X_i=x_k} \sum_{P_i=pa_j} N_{ijk} \log \theta_{ijk} - \frac{\log N}{2} \mathrm{Dim}(\Theta_{Xi|P_i})$$

with N_{ijk} the occurrence number of $\{X_i = x_k$ and $P_i = pa_j\}$ in $\mathcal{D}$.

An improvement of the greedy search in DAG space over CPDAG space have also been proposed by (Chickering, 2002b). The MWST principle is rather different. This algorithm determines the best tree that links all the variables, using a mutual information measurement (Chow and Liu, 1968) or the BIC score variation when two variables become linked (Heckerman et al., 1994).

The aim is to compare improvement of learning algorithms in these different spaces when the dataset is incomplete and when using two different methods for estimating probability: pairwise deletion (ACA, for available cases analysis) and the Expectation-Maximisation algorithm.

2 Dealing with incomplete data

2.1 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 it is not.

$$\mathcal{D}_m = \{X_i^l / M_{il} = 1\}_{1 \leqslant i \leqslant n, 1 \leqslant l \leqslant N}$$
$$\mathcal{D}_o = \{X_i^l / M_{il} = 0\}_{1 \leqslant i \leqslant n, 1 \leqslant l \leqslant N}$$
$$\mathcal{D}_{co} = \{[X_1^l \dots X_n^l] / [M_{1l} \dots M_{nl}] = [0 \dots 0]\}_{1 \leqslant l \leqslant N}$$

Dealing with missing data depends on their nature. (Rubin, 1976) identified several types of missing data:

- MCAR (Missing Completely 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 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 cases 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 missing data distribution. 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 the EM algorithm. More recently, *bound and collapse* (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 first proposed by (Dempster et al., 1977) and adapted by (Lauritzen, 1995) to the learning of the parameters of a Bayesian network whose 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 Θ^*, it is possible to estimate the probability density of the missing data $P(\mathcal{D}_m|\Theta^*)$ and therefore to calculate $Q(\Theta : \Theta^*)$ expectation of the previous log-likelihood:

$$Q(\Theta : \Theta^*) = E_{\Theta^*} [\log P(\mathcal{D}_o, \mathcal{D}_m|\Theta)] \qquad (3)$$

So $Q(\Theta : \Theta^*)$ is the likelihood expectation of any set of parameters Θ calculated using a distribution of the missing data $P(\mathcal{D}_m|\Theta^*)$. Equation 3 can be re-written as follows:

$$Q(\Theta : \Theta^*) = \sum_{i=1}^{n} \sum_{X_i=x_k} \sum_{P_i=pa_j} N_{ijk}^* \log \theta_{ijk} \qquad (4)$$

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.

We are also interested in pairwise deletion which is a method that uses all available data. Cases are removed when they have missing data on the variables involved in that particular computation. This method is very efficient computationally but it assumes that the data are missing completely at random (MCAR). If not, it introduces a bias.

We are interested in studying the results quality that could be expected using these two methods named EM and ACA.

2.2 Determining structure Θ when data are incomplete.

The main methods for structural learning with incomplete data use the EM principle : *Alternative Model Selection EM* (AMS-EM) (Friedman, 1997) and *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) proposes a structural learning method based on genetic algorithm and MCMC.

2.3 General principle and scoring metric

In practice, to perform a maximisation in the joint space $\{\mathcal{G}, \Theta\}$, we must distinguish these two steps[1] :

$$\mathcal{G}^i = \underset{\mathcal{G}}{\operatorname{argmax}} \; Q(\mathcal{G}, \bullet : \mathcal{G}^{i-1}, \Theta^{i-1}) \qquad (5)$$

$$\Theta^i = \underset{\Theta}{\operatorname{argmax}} \; Q(\mathcal{G}^i, \Theta : \mathcal{G}^{i-1}, \Theta^{i-1}) \qquad (6)$$

where $Q(\mathcal{G}, \Theta : \mathcal{G}^*, \Theta^*)$ is the expectation of the likelihood of any Bayesian network $< \mathcal{G}, \Theta >$ calculated using a distribution of the missing data $P(\mathcal{D}_m | \mathcal{G}^*, \Theta^*)$.

Note that the first search (equation 5) 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 possible to look for a better solution to function Q, 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 neighbours of graph $\mathcal{G}$ that have been generated by removal, addition or inversion of an arc interpreted either in the DAG space or in the CPDAG space.

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 calculate marginal likelihood when data are incomplete, so that it is necessary to rely on an efficient approximation like those reviewed by (Chickering and Heckerman, 1996). In complete data cases, the most frequently used measurements are the BIC/MDL score and the Bayesian BDe score. These two scoring metrics are locally consistent but the BIC/MDL score includes a penalty term and we have chosen it for this study.

$$\begin{aligned} Q^{BIC}(\mathcal{G}, \Theta : \mathcal{G}^*, \Theta^*) = \quad & E_{\mathcal{G}^*, \Theta^*}\left[\log P(\mathcal{D}_o, \mathcal{D}_m | \mathcal{G}, \Theta)\right] \\ & -\tfrac{1}{2}\mathrm{Dim}(\mathcal{G}) \log N \end{aligned} \qquad (7)$$

As the BIC score is decomposable, so is Q^{BIC}:

$$Q^{BIC}(\mathcal{G}, \Theta : \mathcal{G}^*, \Theta^*) = \sum_i Q^{bic}(X_i, P_i, \Theta_{Xi|P_i} : \mathcal{G}^*, \Theta^*) \qquad (8)$$

where $Q^{bic}(X_i, P_i, \Theta_{Xi|P_i} : \mathcal{G}^*, \Theta^*) =$

$$\sum_{X_i = x_k} \sum_{P_i = pa_j} N^*_{ijk} \log \theta_{ijk} - \frac{\log N}{2} \mathrm{Dim}(\Theta_{Xi|P_i}) \qquad (9)$$

3 Tested structural learning algorithms

3.1 Greedy Search

3.1.1 SEM

Friedman has proposed two versions of his Bayesian network greedy search algorithm based on suggestions of (Heckerman et al., 1995) but adapted for incomplete datasets : AMS-EM (Friedman, 1997) and BS-EM (Friedman, 1998). We have chosen to use the method AMS-EM that we simply recall SEM as many people do as it could be used with the BIC criterion as explain above.

[1] the notation $Q(\mathcal{G}, \bullet : \dots)$ used in equation 5 stands for $E_\Theta[Q(\mathcal{G}, \Theta : \dots)]$ for Bayesian scores or $Q(\mathcal{G}, \Theta^o : \dots)$ where Θ^o is obtained by likelihood maximisation

3.1.2 GS-ACA

A new implementation a the greedy search inspired by (Cooper and Hersovits, 1992; Heckerman et al., 1994) using pairwise deletion to deal with incomplete dataset is also tested.

3.2 Greedy Equivalent Search

3.2.1 GES-EM

Recent work by (Chickering, 2002a; Castelo and Kocka, 2002; Auvray and Wehenkel, 2002) show that we could take advantage of using the CPDAGs space. Such a space has less equal scoring models than the DAGs space, as many DAGs have the same representation in a unique CPDAG.

A search algorithm in the Markov equivalent space named GES for *Greedy Equivalent Search* has been proposed by (Meek, 1997). It consists in two iterative steps. First one builds iteratively a graph by adding dependence links to the current essential graph (*i.e.* CPDAG) while the second step consists in removing iteratively arcs that are no more needed in the model. The optimality of this method (*conjecture of Meek*) has been proved by (Kočka et al., 2001; Chickering, 2002b).

The GES-EM algorithm keep the principles of the SEM method using the Markov equivalent space to build neighbourhood of the current graph at each steps.

3.2.2 GES-ACA

A version of this method using the available cases analysis (*i.e.* pairwise deletion) is also implemented using the BIC criterion.

3.3 Maximum Weight Spanning Tree

3.3.1 General principle of MWST-EM

(François and Leray, 2004) have shown that, in complete data cases, the MWST algorithm was able to find a simple structure very rapidly (the best tree connecting all the nodes in the space), which could be used as judicious initialisation by the GS algorithm. (Heckerman et al., 1995) suggests using the variation of any decomposable score instead of the mutual information originally used in MWST. Using this remark, we could therefore implement the MWST algorithm using the EM algorithm to manage incomplete datasets.

MWST-EM deals with the choice of the initial structure. The choice of an oriented chain graph linking all the variables proposed by (Friedman, 1997) seems judicious here, since this chain graph also belongs to the tree space. The MWST algorithm used a similarity function between two nodes which is 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 :

$$\left[M_{ij}^{Q}\right] = \Big[Q^{bic}(X_i, P_i = X_j, \Theta_{X_i|X_j} : \mathcal{T}^*, \Theta^*) - Q^{bic}(X_i, P_i = \emptyset, \Theta_{X_i} : \mathcal{T}^*, \Theta^*)\Big] \quad (10)$$

Running maximum (weight) spanning algorithms like Kruskal's or Prim's on matrix M enables us to obtain the best tree that maximises the sum of the local scores on all the nodes, i.e. function Q^{BIC} of equation 8.

This method looks for the best tree-DAG among the neighbours of the current graph. With MWST-EM, we can directly get the best tree that maximises function Q at each step and then this method converge in few steps.

3.3.2 Trees as an initialisation of DAGs greedy search : SEM+T and GS+T-ACA

MWST-EM will serve as initialisation of the SEM algorithms proposed by Friedman. This variant of the structural EM algorithm will be called SEM+T (François, 2006). MWST-ACA will serve as initialisation of the GS-ACA, the resulting method is called GS+T-ACA.

3.3.3 MWST-ACA

We could adapt this algorithm using pairwise deletion. For evaluating the probability $p_{ijk} = \mathbb{P}(X_i = x_{i,k}\, and\, Pa(X_i) = pa_{i,j})$, we no longer need an iterative method as the EM algorithm. To deduce $N_{ijk} = p_{ijk} \times N$, we need to evaluate p_{ijk} on a new sub-dataset that contain only the complete cases of the variables $X_i \cup Pa(X_i)$.

This method is the only direct method to learn a Bayesian network from incomplete dataset in our knowledge.

3.3.4 Extension to classification problems : TAN-EM and TAN-ACA

For classification tasks (where data are incomplete), many studies like those of (Keogh and Pazzani, 1999; Leray and François, 2004b) use a structure based on an augmented naive Bayesian network, where observations (i.e. all the variables except class) are linked to the very best tree (*TAN, Tree Augmented Naive Bayes*). (Geiger, 1992) showed it was the tree obtained by running MWST on the observations. It is therefore possible to extend this specific structure to classification problems when data are incomplete by running a specific version MWST-EM where the class node is considered as a parent of each other nodes. This algorithm will be called TAN-EM.

A version of this method using pairwise deletion named TAN-ACA is also tested.

3.4 Experimental tests

3.4.1 Datasets and evaluation techniques

The experiment stage aims at evaluating all these structure learning methods on incomplete datasets: `Hepatitis`, `Horse`, `House`, `Mushrooms` and `Thyroid` (Blake and Merz, 1998).

We indicate classification rates obtained by the best run on three of the different methods as well as the likelihood and the learning time of the best model on these 3 runs. We also give an 95%-confidence interval based on equation 11 for each classification rate based on (Bennani and Bossaert, 1996).

$$I(\alpha, N) = \frac{T + \frac{Z_\alpha^2}{2N} \pm Z_\alpha \sqrt{\frac{T(1-T)}{N} + \frac{Z_\alpha^2}{4N^2}}}{1 + \frac{Z_\alpha^2}{N}} \quad (11)$$

where N is the sample size, T is the classifier good classification percentage and $Z_\alpha = 1.96$ for $\alpha = 95\%$.

All implementation were done with the *Structure Learning Package* (Leray and François, 2004a) for the *Bayes Net Toolbox* (Murphy, 2001).

3.4.2 Results and interpretations

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. Whilst all runs of Naive Bayes classifier and ACA methods give same results, EM methods do not always give same results because of the first parameters estimation random initialisation. We have also noticed (not reported here) that TAN methods seem the stabler methods concerning the evaluated classification rate while MWST methods seem to be the less ones.

The method GS-EM could obtain very good structures. Then, initialising it with the results of MWST-EM gives stabler results (see (Leray and François, 2005) for a more specific study of this point).

In our tests, except for `Hepatitis` dataset that have only 90 learning samples, TAN methods always obtain structures that lead to better classification rates in comparison with the other structure learning methods.

Remark that MWST methods could occasionally give good classification rates even if the class node is connected to a maximum of two other attributes. In that case, it could be a good hint of most relevant attributes to the class node.

Regarding the log-likelihood reported in table 1, we see that GS-ACA give best results while TAN methods 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.

In these experiments, we could confirm that ACA methods could outperform EM methods on classification for GS and GES learning methods but not systematically. Results are similar for MWST and TAN methods for classification but ACA leads to better log-likelihoods. Classification rates are different but ACA methods could beat EM methods as often as EM methods could beat ACA methods for these two algorithms.

Finally, the table 1 illustrates that TAN and MWST methods have about the same complexity (regarding the computational time) and are a good compromise between Naive Bayes classifiers and Greedy Searches either in DAGs or CPDAG spaces.

Method	HEPATITIS	HORSE	HOUSE	MUSHROOMS	THYROID
sizes	20; 90; 65; 8%	28; 300; 300; 88%	17; 290; 145; 46%	23; 5416; 2708; 31%	22; 2800; 972; 30%
NB	**73.8%** [62.0;83.0]	73.5% [62.0;82.6]	89.7% [83.6;93.6]	94.4% [93.5;95.2]	**96.0%** [94.6;97.1]
	-1122 (0s)	-1540 (0s)	-1404 (0s)	-41147 (0s)	-15728 (0s)
MWST-ACA	58.5% [46.3;69.6]	**82.4%** [71.6;89.6]	90.3% [84.4;94.2]	75.0% [73.3;76.6]	77.4% [74.6;79.9]
	-847 (2s)	-1240 (**16s**)	-1282 (5s)	-31447 (178s)	**-15359** (96s)
MWST-EM	**75.4%** [63.7;84.2]	**82.4%** [71.6;89.6]	82.1% [75.0;87.5]	60.3% [58.5;62.2]	93.8% [92.1;95.2]
	-1114 (**45s**)	-1306 (299s)	-1462 (67s)	-39773 (1389s)	-16912 (2254s)
TAN-ACA	64.6% [52.5;75.1]	73.5% [62.0;82.6]	**93.1%** [87.8;96.2]	**98.4%** [97.8;98.8]	**95.9%** [94.4;97.0]
	-1123 (2s)	-1319 (15s)	-1284 (**4s**)	**-20453** (**183s**)	-15894 (**86s**)
TAN-EM	64.6% [52.5;75.1]	77.9% [66.7;86.2]	91.7% [86.1;95.2]	**98.4**% [97.8;98.8]	**97.0%** [95.7;97.9]
	-1186 (71s)	-1546 (307s)	-1339 (185s)	-33885 (2345s)	-16292 (1936s)
GS-ACA	67.7% [55.6;77.8]	**80.9%** [70.0;88.5]	91.7% [86.1;95.2]	76.7% [75.0;78.2]	77.4% [74.6;79.9]
	-865 (55s)	**-1052** (774s)	-1289 (71s)	-25256 (9086s)	-15394 (2537s)
SEM	64.6% [52.5;75.1]	51.5% [39.8;62.9]	67.6% [59.6;74.7]	74.9% [73.2;76.5]	93.8% [92.1;95.2]
	-1091 (156s)	-1442 (977s)	-1483 (982s)	-50969 (22562s)	-16197 (963s)
GS+T-ACA	58.5% [46.3;69.6]	77.9% [66.7;86.2]	**93.1%** [87.8;96.2]	77.1% [75.5;78.6]	77.4% [74.6;79.9]
	-826 (16s)	**-1052** (603s)	**-1233** (52s)	**-20469** (5050s)	-15391 (856s)
SEM+T	64.6% [52.5;75.1]	51.5% [39.8;62.9]	**93.1%** [87.8;96.2]	74.9% [73.2;76.5]	93.8% [92.1;95.2]
	-1112 (341s)	-1447 (2190s)	-1485 (1094s)	-50969 (30417s)	-15729 (5492s)
GES-ACA	64.6% [52.5;75.1]	**82.4%** [71.6;89.6]	**93.8%** [88.6;96.7]	77.1% [75.5;78.6]	**96.1%** [94.7;97.1]
	-866 (76s)	-1160 (536s)	-1293 (123s)	-23462 (6350s)	-15535 (515s)
GES-EM	64.6% [52.5;75.1]	51.5% [39.8;62.9]	68.3% [60.3;75.3]	74.9% [73.2;76.5]	93.8% [92.1;95.2]
	-1101 (240s)	-1446 (1120s)	-1522 (1062s)	-38947 (54748s)	-16197 (1545s)

Table 1: Two first lines *: dataset names; number of attributes; dataset length; test dataset length; percentage of incomplete entries.* Following lines *: method names; best good classification percentage on three runs; 95%-confidence interval; selected model likelihood; learning time in seconds on a laptop 2.4GHz with Matlab®R2006a.*

4 Conclusions and prospects

Bayesian networks are a tool of choice for reasoning in uncertainty. However, most of the time, Bayesian network structural learning only deal with complete data.

Usually EM principle is used for structure learning as it has been proved to be optimal when the dataset is MAR as ACA is known to introduce a bias when the dataset is MAR and not only MCAR. In those experiments, we have supposed that learning datasets are NMAR as they are real life issued datasets and as it is difficult to know the kind of the dataset if you have not artificially built it. There is no more reasons to use EM rather than ACA during the learning process as both methods are biased.

In this study, ACA (available cases analysis or pairwise deletion) has empirically been compared to EM for Bayesian structure learning. First results show that this method is quite efficient and not very complex. By using it, it is possible to find structures which have a good likelihood and lead to good classification rates, and to do really less time than using the EM algorithm. This first conclusive experiment stage is not final. We are now planning to test and evaluate these algorithms on a wider range of problems.

Moreover, we know the limitation of the BIC criterion and we need to try other criterions: some specific to classification problems as an adaptation of the classification likelihood LCL/LL_c or of ICL to structure learning or more general ones as AIC, AICc, BDe, BDeu, BDγ, MDL/IMDL to study which one perform well with ACA or EM as we have noticed big graphical differences in learnt structures depending on the used method (not reported here because of space limitation).

Acknowledgements

We would like to thank Philippe Leray for the useful discussions we have had.

References

[Auvray and Wehenkel2002] V. Auvray and L. Wehenkel. 2002. On the construction of the inclusion boundary neighbourhood for markov equivalence classes of bayesian network structures. In Adnan Darwiche and Nir Friedman, editors, *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 26–35, S.F., Cal. Morgan Kaufmann Publishers.

[Bennani and Bossaert1996] Y. Bennani and F. Bossaert. 1996. Predictive neural networks for traffic disturbance detection in the telephone network. In *Proceedings of IMACS-CESA'96*, Lille, France.

[Blake and Merz1998] C.L. Blake and C.J. Merz. 1998. UCI repository of machine learning databases.

[Castelo and Kocka2002] R. Castelo and T. Kocka. 2002. Towards an inclusion driven learning of bayesian networks. Technical Report UU-CS-2002-05, Institute of information and computing sciences, University of Utrecht.

[Chickering and Heckerman1996] 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.

[Chickering and Meek2002] 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.

[Chickering et al.1995] D. Chickering, D. Geiger, and D. Heckerman. 1995. Learning bayesian networks: Search methods and experimental results. In *In Proceedings of Fifth Conference on Artificial Intelligence and Statistics*, pages 112–128.

[Chickering2002a] D.M. Chickering. 2002a. Learning equivalence classes of bayesian-network structures. *Journal of machine learning research*, 2:445–498.

[Chickering2002b] D.M. Chickering. 2002b. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, November.

[Chow and Liu1968] C.K. Chow and C.N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

[Cooper and Hersovits1992] G. Cooper and E. Hersovits. 1992. A bayesian method for the induction of probabilistic networks from data. *Maching Learning*, 9:309–347.

[Dash and Druzdzel2003] D. Dash and M.J. Druzdzel. 2003. Robust independence testing for constraint-based learning of causal structure. Proceedings of The Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03), pp 167-174.

[Dempster et al.1977] 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.

[François and Leray2004] O. François and P. Leray. 2004. Evaluation d'algorithmes d'apprentissage de structure pour les réseaux bayésiens. In *14ieme Congrès francophone de Reconnaissance des formes et d'Intelligence artificielle*, pages 1453–1460.

[François and Leray2006] O.C.H. François and P. Leray. 2006. Learning the tree augmented naive bayes classifier from incomplete datasets. In *Proceedings of the Third European Workshop on Probabilistic Graphical Models (PGM'06)*, pages 91–98, Prague, Czech Republic, september.

[François2006] Olivier François. 2006. *De l'identification de structure de réseaux bayésiens à la reconnaissance de formes à partir d'informations complètes ou incomplètes*. Ph.D. thesis, Institut National des Sciences Appliquées de Rouen (INSA), http://ofrancois.tuxfamily.org/these.html.

[Friedman1997] 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.

[Friedman1998] 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.

[Geiger1992] 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.

[Geman and Geman1984] 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.

[Gillispie and Lemieux2001] S.B. Gillispie and C. Lemieux. 2001. Enumerating markov equivalence classes of acyclic digraph models. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001)*, pages 171–177, San Francisco, CA. Morgan Kaufmann Publishers.

[Heckerman et al.1994] D. Heckerman, D. Geiger, and M. Chickering. 1994. Learning Bayesian networks: The combination of knowledge and statistical data. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 293–301, San Francisco, CA, USA, July. Morgan Kaufmann Publishers.

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

[Jensen1996] F.V. Jensen. 1996. *An introduction to Bayesian Networks*. Taylor and Francis, London, United Kingdom.

[Keogh and Pazzani1999] E. Keogh and M. Pazzani. 1999. Learning augmented bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, pages 225–230.

[Kim and Pearl1987] J.H. Kim and J. Pearl. 1987. Convice; a conversational inference consolidation engine. *IEEE Trans. on Systems, Man and Cybernetics*, 17:120–132.

[Kočka et al.2001] T. Kočka, R.R. Bouckaert, and M. Studenỳ. 2001. On characterization inclusion of bayesian networks. In J Breese and D. Koller, editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence.*, pages 261–268. Morgan Kaufmann.

[Lauritzen1995] S. Lauritzen. 1995. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201.

[Leray and François2004a] P. Leray and O. François. 2004a. BNT structure learning package: Documentation and experiments. Technical Report 2004/PhLOF, Laboratoire PSI, INSA de Rouen.

[Leray and François2004b] P. Leray and O. François. 2004b. Réseaux bayésiens pour la classification - méthodologie et illustration dans le cadre du diagnostic médical. *Revue d'Intelligence Artificielle*, 18(2/2004):169–193.

[Leray and François2005] 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.

[Meek1997] C. Meek. 1997. *Graphical Models: Selecting causal and statistical models*. Ph.D. thesis, Carnegie Mellon University.

[Murphy2001] K. Murphy. 2001. The BayesNet Toolbox for Matlab. Computing Science and Statistics: Proceedings of Interface, 33.

[Myers et al.1999] J.W. Myers, K.B. Laskey, and T.S. Lewitt. 1999. Learning bayesian network from incomplete data with stochastic search algorithms. In *the Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI99).*

[Naïm et al.2004] P. Naïm, P.-H. Wuillemin, P. Leray, O. Pourret, and A. Becker. 2004. *Réseaux bayésiens*. Eyrolles, ISBN : 2-212-11137-1.

[Pearl1998] J. Pearl. 1998. Graphical models for probabilistic and causal reasoning. In Dov M. Gabbay and Philippe Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 1: Quantified Representation of Uncertainty and Imprecision*, pages 367–389. Kluwer Academic Publishers, Dordrecht.

[Ramoni and Sebastiani1998] M. Ramoni and P. Sebastiani. 1998. Parameter estimation in Bayesian networks from incomplete databases. *Intelligent Data Analysis*, 2:139–160.

[Ramoni and Sebastiani2000] M. Ramoni and P. Sebastiani. 2000. Robust learning with missing data. *Machine Learning*, 45:147–170.

[Rubin1976] D.B. Rubin. 1976. Inference and missing data. *Biometrika*, 63:581–592.

[Spiegelhalter and Lauritzen1990] D. J. Spiegelhalter and S. L. Lauritzen. 1990. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605.

Robust Classification using Mixtures of Dependency Networks

José A. Gámez[1] and Juan L. Mateo[1] and Thomas D. Nielsen[2] and José M. Puerta[1]
[1] Computing Systems Department – SIMD $i^3\mathcal{A}$ [2] Department of Computer Science
University of Castilla-La Mancha Aalborg University
02071, Albacete, Spain 9220 Aalborg, Denmark

Abstract

Dependency networks have previously been proposed as alternatives to e.g. Bayesian networks by supporting fast algorithms for automatic learning. Recently dependency networks have also been proposed as classification models, but as with e.g. general probabilistic inference, the reported speed-ups are often obtained at the expense of accuracy. In this paper we try to address this issue through the use of mixtures of dependency networks. To reduce learning time and improve robustness when dealing with data sparse classes, we outline methods for reusing calculations across mixture components. Finally, the proposed model is empirically compared to other state-of-the-art classifiers, both in terms of accuracy and learning time.

1 Introduction

Classification is a typical data mining task in which the class label for new instances must be inferred from the values taken by their predictive attributes. The induction of accurate classifiers from pre-labelled data is a hot area of research in machine learning and artificial intelligence. Among the wide range of paradigms used to induce classifiers, Bayesian network classifiers (BNCs) (Friedman et al., 1997) have received much attention.

In this paper we consider the use of a special class of dependency networks (Heckerman et al., 2000) for classification. Compared to BNs, dependency networks allow directed cycles, and structural learning can therefore be performed very efficiently, since the node families can be learned independently of each other. On the other hand, due to the presence of cycles, standard BN inference algorithms cannot be adapted and so approximate algorithms (Gibbs sampling being the most prevalent) are often required. Fortunately, for classification problems where only the class variable is unknown, inference boils down to simply multiplying the appropriate table entries, and it can therefore be performed in linear time w.r.t. the number of variables.

The efficient learning algorithms for DNs hide their main problem, namely inconsistency (a DN does not necessarily represent a joint probability distribution). Moreover, inherent to DNs is the problem of overfitting: the node families are generally larger than in BNs, hence the estimation of the local probability distributions is less reliable and requires more data than in a BN (to reduce parameter variance). To address this problem, Heckerman et al. (2000) proposed to use probabilistic decision trees (PDTs) for parameter specification, and recently Gámez et al. (2008a) has proposed a specification language based on combinations of probability tables (PTs).

A study on inconsistencies in DNs was conducted by Gámez et al. (2008b), who also proposed a heuristic procedure to reduce the inconsistencies. In the study, Gámez et al. (2008b) conclude that PTs outperform PDTs in terms of accuracy, and in their experimental results the inconsistencies almost disappear when using PTs. On the other hand, PDTs have the ability to represent *contextual independence* (i.e., independence relations between variables that hold for some but not necessarily all values), but by focusing on PTs we lose this property.

In order to include aspects of contextual independence when working with PTs, we consider DN-based classifiers inspired by multinets (Geiger and Heckerman, 1996). Multinets are useful for representing certain types of contextual independence, which cannot easily be represented by a single PT-based model. In classification we may, for example, represent different independence relations for the different class values.

As we will also demonstrate, multinets support a notion of *re-usability*. The underlying idea of re-usability is to exploit similarities in the dependency structures across classes. That is, if we have an unbalanced class distribution, we may in some situations be able to reuse parts of a learned probability model (for an instance-rich class) when learning the probability model for a class with few instances. Thus, re-usability may produce more robust classifiers when dealing with unbalanced class distributions.

2 Dependency Networks

Dependency networks (DNs) were proposed by Heckerman et al. (2000) as an alternative to BNs. Formally, a dependency network is a tuple $(\mathbb{G}, \mathbf{P})$ over a domain $\mathbf{X}$ where $\mathbb{G}$ is a directed graph (not necessarily acyclic) and $\mathbf{P}$ is a set of conditional probability distributions, one for each variable in $\mathbf{X}$. Every $P \in \mathbf{P}$ must be such that

$$P(X_i|\mathbf{Pa}_i) = P(X_i|\mathbf{X} \setminus X_i).$$

This means that the set of parents $Pa(X_i)$ for every variable X_i is its Markov blanket $MB(X_i)$.

This definition requires specification *consistency*, in the sense that the joint probability distribution for $\mathbf{X}$ can be recovered from $\mathbf{P}$. This is a very restrictive condition when learning from data, so Heckerman et al. (2000) define *general dependency networks* that relax the factorization requirement by letting $P(\mathbf{X}) \approx \prod_i P(X_i|Pa_i)$.

Since directed cycles are allowed, a DN can be learned from data by learning the parent set for each variable independently, thus supporting fast learning algorithms. On the other hand, by having directed cycles we cannot adapt traditional BN inference algorithms. Instead, Heckerman et al. (2000) relies on approximate inference carried out using Gibbs sampling (Geman and Geman, 1984), and propose the so-called *modified ordered Gibbs sampler*, which is more efficient than standard Gibbs sampling. Fortunately, sampling algorithms are not required when we have complete data for the predictive attributes, and we shall therefore not discuss this topic further.

3 Multinets and Re-usability: Proposed Scheme

Multinets (Geiger and Heckerman, 1996) are useful for representing natural contextual independence assertions. For example, in a classification context we may directly represent the (possibly different) independence relations over the predictive attributes for each of the class values. Consequently a multinet classifier is expected to have a higher, or at least the same, representational power than that of a single BN classifier.

A multinet classifier based on DNs (termed a *MultiDN*) is built by learning a DN model for each class value. Every DN model is built by independently learning the MB for each variable using e.g. the IAMB algorithm (Tsamardinos et al., 2003), specified in Figure 1. The algorithm relies on a method for testing independence, and in this paper we have considered traditional statistical tests, such as G^2, as well as tests measuring the score difference between candidate structures (Chickering, 2002) using e.g. the BIC (Schwarz, 1978) or the BD score (Cooper and Herskovits, 1992).

It should be noted that as an alternative to IAMB, Peña et al. (2007) proposed the PCMB algorithm, which they showed to be more data efficient than IAMB. The results, however, also indicate that for very small data sets (e.g. Alarm with 100 instances) PCMB has worse precision (although better recall) than IAMB. The combination of these two factors means that PCMB may identify larger candidate MB sets than IAMB. This behavior was also confirmed in our

```
1  { Phase I (forward) }
2  MB = ∅
3  While MB has changed
4     Y = argmax_{X∈U\(MB∪{T})} dep(X, T|MB)
5     If Y ⫫̸ T|MB Then
6        MB = MB ∪ {Y}
7
8  { Phase II (backwards) }
9  For each X ∈ MB Do
10    If X ⫫ T|(MB\X) Then
11       MB = MB\{X}
12
13 Return MB
```

Figure 1: The incremental association Markov blanket (IAMB) algorithm for learning the MB for a target variable T.

preliminary experiments, where we compared both algorithms based on the datasets listed in Section 4 (having similar ratio between the number of variables and instances as the example above). This limits the usability of PCMB for learning DNs, since the larger MBs make the probability estimates less reliable.

3.1 Re-usability

In a multinet classifier we basically need to learn several networks, one for each class value. Assuming that the set of independence statements for the different class values are not disjoint, one might be able to use parts of the learned probability model for one class value when learning the probability model for another class value. The potential advantages are twofold: First we may speed up learning, and, secondly, we may obtain a more robust classifier when data is scarce for some of the classes.

For MultiDNs, re-usability consists in seeding the MB learning algorithm with a candidate MB set. In the IAMB algorithm this is achieved by simply replacing line 2 with **MB = seedMB**. We call this new algorithm *SeededIAMB*. If the candidate seed is good, i.e., it corresponds to the true MB or a subset hereof, the algorithm can achieve substantial computational savings (the situation where this is not the case is discussed below).

Theorem 1. *Under the assumptions that the independence tests are correct and that the database D is an independent and identically distributed sample from a probability distribution P faithful to a DAG $\mathcal{G}$, SeededIAMB identifies the true MB for the target variable.*

Proof sketch: Given any initial set the algorithm will in the forward phase always introduce all variables in the MB because, by definition, there is no set of variables that can make the variables in the MB independent of the target. So at the end of this phase we will have a super-set of the MB for the target variable. The backward phase, removing all false positives, behaves in the same way as the non-seeded version. □

As indicated above, if the candidate set used for *SeededIAMB* is close to the actual MB, then we may get computational savings. On the other hand, if the candidate set does not intersect the true MB, then the seeded version of the IAMB algorithm may introduce a computational overhead. Thus, it is important to find a good ordering in which to process the classes, and to be able to determine when a previous structure should be used as seed. Below we have detailed some of our considerations.

• Ordering the class values: In our experiments we order the classes according to the number of instances associated with each class value, starting with the class value having most instances. Here we assume that more data yields more reliable MB estimates for potential reuse.

• Determine the score for a candidate seed structure: Assume that we have a collection of previous models $DN_i, i = \{1, \ldots, n\}$, and that the current model must be learned from data $\mathcal{D}_{c_j}$. The log-likelihood of the previous models DN_i given $\mathcal{D}_{c_j}$, $\log L(DN_i|\mathcal{D}_{c_j})$, can be used to select which of these models to use as seed. Furthermore, we can also use the local score for each single variable X

$$\frac{1}{N_j}\sum_{l=1}^{N_j}\log P(X|MB(X)_i, DN_i)(\mathbf{d}_l),$$

indicating how well $MB(X)_i$ predicts X in $\mathcal{D}_{c_j}$.

• Setting the threshold for when to reuse a MB: A possible threshold value is the score

of the empty structure, since if IAMB is not seeded, then this is the structure that is used as prior. Here we have two strategies. One strategy (called *BESTlogL*) consists in picking the best $MB(X)_i$ for a given variable X, if its score is greater than that of the empty MB. As another strategy (called *THRESHOLDlogL*), we can pick the union of all $MB(X)_i$ for all previous models that have greater score than the empty MB.

• When to compute the score of a candidate seed structure: Although the log-likelihood is easy to compute (its complexity is linear in the number of instances), it still incurs an overhead for the algorithm. We have considered three alternatives for deciding on re-usability without having to compute a score; thus, saving computation time but making the selection less informed. The first method simply uses the first learned model as seed for all the subsequent models, and is labelled *First*. The second method uses the intersection of the MBs for all the previous learned models (*Intersection*), and the third method uses the union of the MBs for all the previous learned models (*Union*).

4 Experiments

We have performed a set of experiments in order to analyze the performance of the proposed algorithm in terms of classification accuracy, learning time, and the potential improvement obtained by re-usability.[1] For evaluating the accuracy and learning time, we have compared our classifier with a collection of other well-known classifiers, both probabilistic and non-probabilistic. For the actual learning, we have used 28 dataset from the UCI repository (Asuncion and Newman, 2007), see Table 1.

All experiments have been carried out on a PC with a 3GHz Intel Pentium IV processor and 2Gb RAM memory.

[1]Given the space limitation not all results of our experiments are included, but a complete list can be found at http://www.dsi.uclm.es/personal/JuanLuisMateo/mixtureDN.html

Table 1: Datasets used in our experiments.

dataset	insts.	vars.	|class|
australian	690	14	2
autos	205	23	7
balance	625	5	3
breast-cancer	286	10	2
breast-w	699	10	2
car	1728	7	4
cmc	1473	10	3
diabetes	768	7	2
ecoli	336	7	8
heart	270	14	2
hepatitis	155	20	2
ionosphere	351	34	2
iris	150	5	3
kr-vs-kp	3196	37	2
labor	57	12	2
mushroom	8124	23	2
nursery	12960	9	5
page-blocks	5473	11	5
post-op	90	9	3
segment	2310	20	7
soybean	683	36	19
spambase	4601	56	2
vehicle	846	19	4
vote	435	17	2
vowel	990	14	11
waveform	5000	20	3
wine	178	14	3
zoo	101	17	7

4.1 Algorithms

The algorithms used in the comparison are J48 (Quinlan, 1993), multilayer perceptron (NN) (Bishop, 1995), k-nearest neigbours (kNN) (Aha and Kibler, 1991), support vector machine (SVM) (Platt, 1999), naive Bayes (NB) (Langley et al., 1992), k-dependence Bayesian classifier (kDB) (Sahami, 1996), tree augmented naive Bayes (TAN) (Friedman et al., 1997), multinet with Bayesian networks (MultiBN) (Friedman et al., 1997), and another DN-based classifier (ChiSqDN) (Gámez et al., 2006).

We have used the Weka implementation of J48, NN, kNN, and SVM (Witten and Frank, 2005). For these classifiers, all parameters were set to their default values, except for kNN for which we tried both $k = 1$, and $k = 3$ with inverse distance weighting. For kDB we have experimented with $k = 1, 2, 3, 4$. For MultiBN we have considered two variants. One based on the PC learning algorithm (Spirtes et al., 2001), and the other based on local search (hill-

climbing) with the BIC (Schwarz, 1978) or the BD score (Cooper and Herskovits, 1992). For MultiDN we used the IAMB algorithm to determine the MB for each variable, and the independence tests were performed using either G^2 or by measuring the difference in BIC score for the candidate structures. All algorithms (except J48, NN, kNN and SVM) have been implemented in Java with the Elvira software (Elvira Consortium, 2002). Accuracy is assessed using a 5x2 cross validation scheme.

4.2 Results

For each algorithm appearing in several versions (different parameter settings) we only report on the version giving the best accuracy results. That is, kNN with $k = 3$ and inverse distance weighting, kDB with $k = 1$, and multinets based on the BIC score.

Table 2 shows the accuracy results for some of the selected classifiers; the results reported for MultiDN relates to the BIC variant with no reuse across classes. Due to space restrictions we have only included the best classifiers, but the results for the remaining classifiers can be found at the web page specified above.

From Table 2 we see that SVM achieves the best result on average. In order to determine which of the other classifiers that (from a statistical point of view) cannot be considered weaker than SVM, we have carried out Holm's post-hoc test with the SVM classifier as control. This test shows that kDB and both multinet-based classifiers are comparable with SVM; the remaining classifiers receive worse results and the differences are statistically significant.

4.2.1 Learning Time

In Table 3 we list the learning time for the algorithms selected above. MultiDN obtains the best results in several cases and is never the worst. MultiBN, on the other hand, is never the best and several times the worst, whereas SVM is the fastest for most of the datasets, but it is sometimes the slowest too. In order to test whether there is a statistical difference, we have evaluated the results using Wilcoxon's signed rank test. With significance level 0.05 the test indicates that MultiDN is significantly faster than kDB and MultiBN, but there is no such difference between SVM and MultiDN, and they can therefore be considered equally fast. The critical values for the comparisons are 1.42e-3 for kDB, 8.20e-7 for MultiBN, and 0.52 for SVM.

Table 2: Accuracy results. For each dataset, the best results are shown in bold face, and the worst results are underlined.

	kDB	MultiBN	MultiDN	SVM
australian	84.64	85.42	**86.35**	84.93
autos	79.02	79.81	73.27	**82.14**
balance	74.05	73.82	74.08	74.11
breast-cancer	70.28	69.79	70.49	71.12
breast-w	96.22	96.57	97.34	96.68
car	93.26	91.68	91.01	92.45
cmc	**53.96**	52.49	53.29	53.96
diabetes	77.47	77.68	**79.27**	76.77
ecoli	84.82	85.12	82.26	83.87
heart	81.63	80.3	82.37	83.7
hepatitis	**87.88**	86.45	85.81	85.55
ionosphere	91.97	**92.65**	92.19	90.48
iris	95.07	94.53	96.27	**96.40**
kr-vs-kp	94.22	96.49	95.27	95.24
labor	89.8	92.22	**94.72**	92.29
mushroom	99.87	**100.00**	99.95	99.99
nursery	93.26	95.57	93.73	93.06
page-block	95.75	96.42	96.24	**96.81**
post-op	66.22	66.89	66.89	69.56
segment	94.17	94.94	91.36	95.56
soybeam	91.6	**94.00**	93.35	92.5
spam-base	92.73	93.65	92.58	**93.81**
vehicle	71.23	71.28	70.33	**73.14**
vote	93.75	93.89	94.16	**95.22**
vowel	73.17	69.82	64.99	79.07
waveform	82.25	81.79	81.26	**84.86**
wine	97.08	97.98	98.31	97.87
zoo	94.65	94.26	94.06	94.26
aver.	85.72	85.91	85.4	**86.62**

4.2.2 Re-usability Analysis

To get an indication of the theoretical complexity of the re-use methods, we introduce the notion of reuse-complexity, which is defined as the number of computations[2] times the average number of variables involved in each computation; note that reuse-complexity does not take into account the overload introduced by *BESTlogL* and *TRHESHOLDlogL*. This overload is considered in learning time. Table 4 shows the average complexity and learning time among all datasets.

From the results we see that reusability based on *Intersection* achieves the best results in

[2]A computation is a call to the function that either calculates the score of a local structure with BIC or BDe, or performs a statistical test with G^2.

Table 3: Learning time in mileseconds.

	kDB	MultiBN	MultiDN	SVM
australian	543	1036	249	**174**
autos	711	3226	951	**268**
balance	**30**	46	37	85
breast-cancer	89	372	94	**72**
breast-w	196	228	130	**68**
car	173	166	**114**	352
cmc	396	428	**281**	653
diabetes	83	96	**67**	81
ecoli	**42**	59	63	325
heart	213	256	129	**40**
hepatitis	348	433	254	**33**
ionosphere	3995	5716	1849	**105**
iris	**14**	22	17	60
kr-vs-kp	47422	63283	18906	**1545**
labor	41	75	62	**29**
mushroom	25730	93243	13727	**3792**
nursery	2481	2087	**1285**	17293
page-block	1960	4760	**1443**	2763
post-op	**29**	58	37	61
segment	4875	14380	**2343**	2863
soybeam	8425	4374	4383	**2651**
spam-base	217476	79629796	39136	**8311**
vehicle	1557	15215	1428	**653**
vote	573	847	389	**49**
vowel	**792**	8088	994	1582
waveform	10810	6131	**3815**	10478
wine	144	148	122	**66**
zoo	**157**	166	217	303

terms of both complexity and learning time. Moreover, both *BESTlogL* and *TRHESHOLDlogL* have the highest learning times, which indicates that their overload for determining when and what to re-use is not justified by the learning time. As part of future work, we plan to investigate other ways to evaluate candidate seed structures.

Table 4: Complexity and learning time for the different re-usability schemes averaged over all datasets.

	Complexity	Time
NOREUSE	2045	3303
BESTlogL	1920	3671
TRHESHOLDlogL	1925	3861
First	1915	3201
Intersection	**1899**	**3136**
Union	1945	3400

The underlying idea of reusability is to use the probability estimates from data-rich classes to improve the estimates for data-poor classes. In order to evaluate this idea we have selected a subset of the datasets in Table 1, all of which have an unbalanced class distribution. The results can be found in Table 5, which shows the absolute difference in accuracy between each of the proposed re-usability methods and the plain algorithm without re-usability. On average we can see that there is always an improvement, and to compare the methods we have carried out a one-tailed Wilcoxon's signed rank test with significance level 0.05. The critical values for these five methods are 0.03 for *BESTlogL*, 0.01 for *THRESHOLDlogL*, 0.02 for *First*, 0.07 for *Intersection*, and 0.13 for *Union*. From these values the experiments indicate that *BESTlogL*, *THRESHOLDlogL*, and *First* significantly improve the original algorithm's performance. The improvement is usually small, but we have to bear in mind that this improvement is typically over class values with few instances.

Table 5: Absolute difference in accuracy for each of the re-usability methods with respect to the plain learning algorithm without re-usability. 1=*BESTlogL*, 2=*THRESHOLDlogL*, 3=*First*, 4=*Intersection*, 5=*Union*

	1	2	3	4	5
autos	1.17	1.07	1.07	-0.20	0.87
balance	0.00	0.00	-0.03	-0.03	0.00
breast-cancer	0.28	0.28	0.28	0.28	0.28
breast-w	0.00	0.00	0.00	0.00	0.00
car	0.00	0.00	0.00	0.00	0.00
cmc	0.19	0.20	0.20	0.11	0.29
diabetes	0.03	0.03	0.08	0.08	0.08
ecoli	0.00	0.00	0.00	0.00	0.00
hepatitis	0.26	0.26	0.65	0.65	0.65
ionosphere	-0.11	-0.11	-0.17	-0.17	-0.17
nursery	0.00	0.00	0.00	0.00	-0.58
page-block	0.04	0.04	0.05	0.04	0.05
post-op	0.00	0.00	0.00	0.00	0.00
soybeam	-0.09	0.03	-0.03	0.06	-0.70
spam-base	0.36	0.36	0.13	0.13	0.13
vote	0.09	0.09	0.23	0.23	0.23
zoo	0.00	0.00	0.00	0.00	0.20
average	0.13	0.13	0.14	0.07	0.08

In order to investigate this aspect further we can consider the confusion matrices for the two datasets cmc and hepatitis. For each dataset we perform learning with re-usability (using the *First* method) and without re-usability (see Tables 6 and 7); for *BESTlogL* and *THRESHOLDlogL* we obtain the same behavior. In these matrices we see an improvement for the class value with fewer instances, which corresponds to the

second column in `cmc` and the first column in `hepatitis`. These states represent 23% and 21% of the instances, respectively, and in both cases the improvement obtained with re-usability is 3% for that state.

The impact of these results can be illustrated by considering e.g. medical diagnosis, where the number of cases with people being sick is typically much smaller than the number of cases with healthy people. A false negative for a person being sick means that she will not be given a treatment for her illness (possibly with disastrous consequences). With re-usability we can reduce this mis-classification rate, which, in situations like the medical example above, can have significant consequence.

Table 6: Confusion matrices without re-usability (a) and using with re-usability using the *First* method (b) on the cmc dataset.

	0	1	2
0	363.8	72.0	127.2
1	82.8	150.2	112.8
2	182.4	110.8	271.0

(a)

	0	1	2
0	360.4	66.4	118.8
1	87.6	159.4	124.0
2	181.0	107.2	268.2

(b)

Table 7: Confusion matrices without re-usability (a) and with re-usability using the *First* method (b) on the hepatitis dataset.

	0	1
0	21.8	11.8
1	10.2	111.2

(a)

	0	1
0	22.6	11.6
1	9.4	111.4

(b)

In comparison with the SVM classifier (see Table 8) we see a significant difference: the SVM classifier has the best average accuracy, but, in these cases, it has difficulties with class values having few instances. This behaviour can be expected by taking into account how this classifier is built. Nonetheless, if we instead look at the MultiBN or kDB classifiers, we also observe (results not included) that the proposed classifier obtains better results for states with poor representation of instances.

5 Conclusions and Future Work

In this paper we have presented a probabilistic classifier based on a class mixture of dependency networks. In addition to supporting fast learning algorithms, the proposed classifier allows for intermediate results to be reused across classes, thereby obtaining potential computational savings as well as improving the robustness for data scarce classes. We have proposed strategies for deciding on what and when to reuse. However, while the preliminary results indicate that the proposed strategies can improve classification accuracy, they also indicate that a simple uninformed strategy achieves better learning time results than more elaborate strategies. Measured in terms of the re-usability percentage, preliminary results indicate that the heuristic *BESTlogL* obtains the best results; for this strategy, 35% of the seeded variables also appear in the final Markov blankets (averaged over all the datasets). Designing other heuristic functions for guiding re-usability is a topic for future research. Here the aim is to find strategies that give a more balanced trade-off between accuracy improvements and learning time overhead.

As part of future work we also plan to conduct more extensive re-usability experiments on larger datasets. Medical datasets, in particular, can be of interest, since they typically have hundreds of variables and relatively few instances. Another issue for future work is how to establish a good class ordering. One may, for example, be able to exploit the natural class ordering found in ordinal variables, i.e., we may expect that class values close to each other induce similar dependency structures over the attributes.

Table 8: Confusion matrices for SVM with cmc (a) and hepatitis (b) datasets.

	0	1	2
0	387.0	81.8	139.8
1	54.8	95.6	59.0
2	187.2	155.6	312.2

(a)

	0	1
0	19.0	9.4
1	13.0	113.6

(b)

Acknowledgments

This work has been partially supported by Spanish Ministerio de Educación y Ciencia (project TIN2007-67418-C03-01); Junta de Comunidades de Castilla-La Mancha (project PBI-

08-048) and FEDER funds.

We would like to thank José Manuel Peña for useful comments and suggestions for earlier versions of this paper. We would also like to thank the anonymous reviewers for their constructive comments.

References

D. Aha and D. Kibler. 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

A. Asuncion and D.J. Newman. 2007. UCI machine learning repository.

C.M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

D.M. Chickering. 2002. Optimal Structure Identification With Greedy Search. *Journal of Machine Learning Research*, 3:507–554.

G.F. Cooper and E. Herskovits. 1992. A Bayesian Method for the Induction of Probabilistic Networks from data. *Machine Learning*, 9(4):309–347.

Elvira Consortium. 2002. Elvira: An Environment for Creating and Using Probabilistic Graphical Models. In *Proceedings of the Fist European Workshop on Probabilistic Graphical Models*, pages 222–230.

N. Friedman, D. Geiger, and M. Goldszmidt. 1997. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163.

J.A. Gámez, J.L. Mateo, and J.M. Puerta. 2006. Dependency networks based classifiers: learning models by using independence test. In *Third European Workshop on Probabilistica Graphical Models (PGM06)*, pages 115–122.

J.A. Gámez, J.L. Mateo, and J.M. Puerta. 2008a. Improved EDNA (Estimation of Dependency Networks Algorithm) Using Combining Function with Bivariate Probability Distributions. In *Genetic and Evolutionary Computation Conference (Gecco08)*.

J.A. Gámez, J.L. Mateo, and J.M. Puerta. 2008b. Towards consistency in general dependency networks. Technical Report DIAB-08-04-1, Computing Systems Deparment, University of Castilla-La Mancha.

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

S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:147–156.

D. Heckerman, D.M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. 2000. Dependency networks for inference, collaborative filtering and data visualization. *Journal of Machine Learning Research*, 1:49–75.

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.

J.M. Peña, R. Nilsson, J. Björkegren, and J. Tegnér. 2007. Towards scalable and data efficient learning of markov boundaries. *International Journal of Approximate Reasoning*, 45(2):211–232.

J. Platt, 1999. *Advances in Kernel Methods - Support Vector Learning*, chapter Fast Training of Support Vector Machines using Sequential Minimal Optimization, pages 185–208. MIT Press.

R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

M. Sahami. 1996. Learning Limited Dependence Bayesian Classifiers. In *Second International Conference on Knowledge Discovery in Databases*, pages 335–338.

G. Schwarz. 1978. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464.

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

I. Tsamardinos, C.F. Aliferis, and A. Statnikov. 2003. Algorithms for large scale markov blanket discovery. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference FLAIRS 2003*.

I.H. Witten and E. Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edition edition.

Towards consistency in general dependency networks

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

Dependency networks are a probabilistic graphical model that claim several advantages from other models like Bayesian networks and Markov networks, for instance. One of these advantages in general dependency networks, which are the object of study in this work, is the ease of learning from data. Nonetheless this easiness is also the cause of its main drawback: inconsistency. A dependency network cannot encode the probability distribution underlay in the data but an approximation. This approximation can be enough good for some applications but not in other cases. In this work we make a study of this inconsistency and propose a method to reduce it. From the conclusions we have taken from this analysis we have developed an algorithm that has to be run after the standard learning algorithm yields its solution. Our method is an heuristic approach so we cannot assure that the resulting model is fully consistent, however we have carried out some experiments which make us to think that it produces high quality models and therefore is advisable its use.

1 Introduction

Probabilistic graphical models (PGM) (Lauritzen, 1996; Jensen and Nielsen, 2007) have been deeply under research and have been used in many applications in the last two decades because of their capabilities. There are several kinds of PGMs like decision graphs or Markov networks (MN), but probably the most famous and used are Bayesian networks (BN).

Dependency networks (DN) are a probabilistic graphical model proposed by (Heckerman et al., 2000) as an alternative to BN. The main difference between them is that the graph in DN does not have to be acyclic. The parametric component is the same, i.e. every variable has a conditional probability distribution given its parents. Another difference is that in DNs the parents for each variable is its *Markov Blanket* (MB) in the Bayesian network encoding the same domain. This is the reason why the graph of a dependency network can be cyclic.

In (Heckerman et al., 2000) are presented some tasks in which DNs can be worthwhile like probabilistic inference, collaborative filtering and visualization of relationships. Nonetheless, from the automatic learning point of view DNs have a drawback because its not easy to learn a set of conditional probability distributions (CPD) consistent with the joint probability distribution (JPD). That is the reason the authors relaxed the definition of DNs and they defined *general dependency networks*, however now we cannot expect that with the set of CPDs we were able to recover the JPD but an approximation. Heckerman et al. (2000) argue that this approximation can be better as the amount of data used in the learning process increases, however it still is an approximation.

In this work we want to analyze how this approximation can deteriorate the performance of a DN model and we propose a way to improve the whole model with a minimun computational cost or even speeding up the learning process.

In Section 2 we present a more formal and detailed definition of DNs. In Section 3 we

make an analysis of the inconsistencies in general DNs. In Section 4 we explain our proposal to reduce those inconsistencies. In Section 5 we describe some experiments we have carried out in order to validate our proposal and show the results, and in Section 6 we conclude.

2 Dependency Networks

2.1 Consistent Dependency Networks

Given a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ with a positive JPD $P(\mathbf{X})$, a *consistent dependency network* for this domain consists of a pair $(\mathcal{G}, \mathcal{P})$ where $\mathcal{G}$ is a directed graph (not necessarily acyclic), in which every node represents a variable, and $\mathcal{P}$ is a set of CPDs. In $\mathcal{G}$ the set of parents for each variable X_i, denoted by $\mathbf{Pa}_i$, is formed by all those variables such that verify

$$P(X_i|\mathbf{Pa}_i) = P(X_i|X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n). \quad (1)$$

So if $P(\mathbf{X})$ is faithful to a graph, what is a common assumption in machine learning algorithms, then the parents for a given variable in a DN are its MB. Other way to say so is that a DN has the same adjacencies than a MN.

A DN is consistent in the sense that all the CPDs in $\mathcal{P}$ can be obtained from the JPD $P(\mathbf{x})$, i.e. we can obtain $P(\mathbf{X})$ from $\mathcal{P}$ in a similar way than in an BN or MN by the product of local probability distributions:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}_i)$$

In (Heckerman et al., 2000) is shown the equivalence between DNs and MNs. The only difference is that in MNs the quantitative component is provided by potential functions whereas in DNs is provided by CPDs. Given this equivalence one approach to learn DNs from data can be to learn a MN in order to obtain the structure and then compute the set of CPDs from the MN via probabilistic inference. Other possibility suggested also in that paper is to learn another probabilistic model (a BN for instance) and translate it into a DN. Nonetheless the problem with these approaches is that the conversion can be computational expensive and inefficient in many cases. That is the reason why the authors presented another definition for DNs more relaxed in order to ease the automatic learning from data. This new definition is covered in next section.

2.2 General Dependency Networks

A consistent DN is not attractive from an machine learning point of view because of the difficulties related with obtaining the set of CPDs, specially with the restriction that this set has to be consistent with the JPD for the variables in the domain. So *general* DNs, also described in (Heckerman et al., 2000), are based on the idea of removing those restriction about consistency. Thus every single CPD $P(X_i|\mathbf{Pa}_i)$ can be estimated independently from the others by any probabilistic classification method, as a probabilistic decision tree (PDT) (Buntine, 1991). Once we have all the CPDs we can build the structure of the dependency network from the (in)dependencies that are appeared during the learning process.

This way of learning a DN can be more efficient than learning from a MN in many cases, and other advantage is that its parallelization is straightforward, what can report a great benefit if we are dealing with a domain with a large number of variables. Nonetheless this heuristic approach has a disadvantage, due mainly to the independent search over the variables and poor estimations in small datasets, the learned CPDs may not be consistent with the JPD, this can be called *parametrical inconsistency*. But also *structural inconsistencies* can appear because after learning the CPD we can see that, for instance, X_i can be parent of X_j but not the opposite, i.e. the CPD for X_i would not contain X_j but the CPD for X_j would contain X_i. In (Heckerman et al., 2000), authors argue that this inconsistencies can be reduced as the amount of data used for leaning increases.

A formal definition for this new model is as follows. Given a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, consider the set of CPDs $\mathcal{P} = \{P_1(X_1|\mathbf{X}\backslash X_1), P_2(X_2|\mathbf{X}\backslash X_2), \ldots, P_n(X_n|\mathbf{X}\backslash X_n))\}$. It is not required that these distributions are consistent with $P(\mathbf{X})$, i.e. it is not required that this set can be obtained via inference from the JPD. Under these conditions a *dependency net-*

work for $\mathbf{X}$ and $\mathcal{P}$ is the pair $(\mathcal{G}, \mathcal{P}')$, where $\mathcal{G}$ is a directed graph usually cyclic and $\mathcal{P}'$ is a set of CPDs such that

$$P_i(X_i|\mathbf{Pa}_i) = P_i(X_i|\mathbf{X}\backslash X_i) \quad (2)$$

for every $P_i \in \mathcal{P}$.

2.3 Inference

In any case, with a consistent or general dependency network, given the likely existence of cycles in the graph we cannot use exact inference algorithms used in BNs and some of the approximate. In the case of consistent DNs it can be converted to a MN and use standard techniques for probabilistic inference over MNs. Nonetheless a more general option is suggested in (Heckerman et al., 2000) for both models, Gibbs sampling (Geman and Geman, 1984). Basically this method works by repeatedly cycling through each variable in a fixed order during all the process, and sampling each X_i according to $P(X_i|\mathbf{Pa}_i)$. This procedure is called *ordered Gibbs sampler* but in the case of a general DN, given that the CPDs may not be consistent with the JPD, is called *ordered pseudo-Gibbs sampler*. Besides in (Heckerman et al., 2000) it is developed a more efficient method which can avoid some sampling and it is called *modified ordered (pseudo-)Gibbs sampler*. This method, in order to get $P(\mathbf{Y}|\mathbf{Z})$ and the value of $\mathbf{Z}$ is $\mathbf{z}$ for a DN in a domain with a set of variables $\mathbf{X}$, is shown in Figure 1.

The key point is in line 6, since if the values for all the parents for a given variable are known we can avoid the sampling for that variable and just take its value from its CPD. This algorithm is justified by Equations (1) and (2).

However, given the modified ordered Gibbs sampler, there are some situations in which we can avoid completely the sampling. For instance if we use a DN as a classifier and we assume that we always know the values for all predictive variables (Gámez et al., 2006). Other case is when is needed to obtain the probability for a full configuration in a DN, i.e. $P(\mathbf{X})$ when we have fixed the value for every single variable. We can see this computation in other way

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{X}\backslash X_i).$$

```
1  U = Y (* the unprocessed variables *)
2  P = Z (* the processed and conditioning
       variables *)
3  p = z (* the values of P *)
4  While U ≠ ∅
5     Pick X_i ∈ U s.t. X_i has no more
         parents in U than any variable
         in U
6     If all parents of X_i are in P
7        P(X_i|p) = p(X_i|Pa_i)
8     Else
9        Use modified ordered Gibbs
            sampling to get P(X_i|p)
10    U = U − X_i
11    P = P + X_i
12    p = p + x_i
13 Return the product of the conditionals
       P(X_i|p)
```

Figure 1: Modified ordered Gibbs sampler

If we take the right part of the equation and use the modified ordered Gibbs sampler we have that $\mathbf{Y} = \{X_i\}$ and $\mathbf{Z} = \{\mathbf{X}\backslash X_i\}$, in line 5 we have only one choice and in line 6 the condition is true so the sampling is avoided. Therefore we can compute statistics such as the likelyhood of a DN for a dataset, and we assume that the dataset does not contain missing data, without any sampling.

3 Analysis of Parametrical Inconsistency

In this section we want to analyze some issues regarding with inconsistency in DNs. From now on we consider only general DNs.

Example 1. Consider the case in which we have two variables, X and Y, and they are dependent, then the DN for this domain, $\mathcal{DN}$, should have a graph with two links $X \rightarrow Y$ and $X \leftarrow Y$.

Hence $\mathcal{P}' = \{P(X|Y), P(Y|X)\}$ for $\mathcal{DN}$. However is clear that $P(X,Y) \neq P(X|Y) \cdot P(Y|X)$. In fact

$$\hat{P}(X,Y) = \frac{P(X,Y) \cdot P(X,Y)}{P(X) \cdot P(Y)} = f(X,Y) \cdot P(X,Y) \quad (3)$$

where

$$f(X,Y) = \frac{P(X,Y)}{P(X) \cdot P(Y)}$$

Therefore, even in a situation so simple like this one, we cannot expect to have a DN without

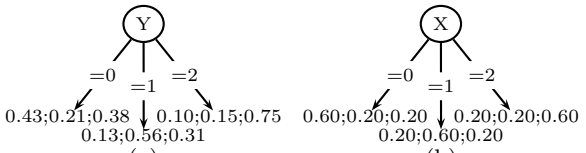
inconsistencies, when it is learned from data of course. Moreover, looking at Equation (3) we can say that the inconsistency is smaller as the dependency between X and Y is weaker and $f(X,Y)$ tends to 1.

In (Heckerman et al., 2000) they propose learn DNs by means of probabilistic decision trees. This model are very good to encode contextual dependencies. Encoding the CPDs by probabilistic decision trees can help to reduce the inconsistencies because a decision tree tries to represent a more general probability distribution by pruning some branches which are similar. Then the dependence between the variables can be smoothed and thus $f(X,Y)$ is closer to 1. However if this happens we have a poorer estimation for the JPD even though is less inconsistent.

In order to illustrate that we can consider both variables in Example 1 are discrete with three states and their JPD is defined by this table:

	Y=0	Y=1	Y=2
X=0	0.12	0.04	0.04
X=1	0.06	0.18	0.06
X=2	0.10	0.10	0.30

When we compute $P(X|Y)$ and $P(Y|X)$ from $P(X,Y)$ in the way of a probability table or a full expanded probabilistic decision trees (see Figure 2 (a) and (b)) we obtain an estimation $\hat{P}_1(X,Y)$ which is shown in Figure 2(c):



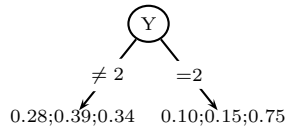
(a) (b)

	Y=0	Y=1	Y=2
X=0	0.257	0.025	0.020
X=1	0.043	0.338	0.030
X=2	0.075	0.063	0.450

(c)

Figure 2: Joint probability distribution $\hat{P}_1(X,Y)$ (c) obtained using full decision trees for CPDs $P(X|Y)$ (a) and $P(Y|X)$ (b).

We can see large differences between the true figures and the estimation. Besides we can check that $\hat{P}_1(X,Y)$ is not a probability distribution because $\sum_{x,y} \hat{P}_1(x,y) = 1.301 \neq 1$. If, for instance, the learning procedure decides to change the representation of $P(X|Y)$ for a probabilistic decision tree in which branches for values 0 and 1 are merged (Figure 3(a)), because they are the most similar, then we obtain a new estimation $\hat{P}_2(X,Y)$ which is shown in Figure 3(b). $\hat{P}_2(X,Y)$ still differs from $P(X,Y)$ but is closer to it than $\hat{P}_1(X,Y)$ in average, and also is closer to be a probability distribution because $\sum_{x,y} \hat{P}_2(x,y) = 1.170$.



(a)

	Y=0	Y=1	Y=2
X=0	0.166	0.055	0.020
X=1	0.078	0.233	0.030
X=2	0.069	0.069	0.450

(b)

Figure 3: Joint probability distribution $\hat{P}_2(X,Y)$ (b) obtained using a simpler decision tree for $P(X|Y)$ (a).

Therefore in spite of the use of probabilistic decision trees we still have an approximation which could not be good enough for some applications. In next section we present a simple heuristic method that can reduce inconsistencies in DNs improving its accuracy.

4 How to Improve Consistency

As it has been seen in the previous Section even in a simple case like Example 1 we cannot expect to get a consistent DN if the CPDs are learned independently. If two variables are dependent this equation $P(X,Y) = P(X|Y) \cdot P(Y|X)$ will never be true, nonetheless this expression $P(X,Y) = P(X) \cdot P(Y|X) = P(X|Y) \cdot P(Y)$ is always true and does not matter if both variable are dependent or not. Bearing this in mind, in Example 1 we can ensure consistency if at the end of the learning process we realize that X is a predictive variable for Y and vice versa and then instead of maintaining both CPDs we replace $P(X|Y)$ by $P(X)$ or $P(Y|X)$ by $P(Y)$. This is the basic idea of our proposal, but there is not so easy when there are more variables in-

volved. In that case we do not expect to obtain the best set of probabilities whose composition yield the right JPD, but a good approximation and, more important, more consistent.

More precisely the proposal consists in estimating a set of CPDs of a BN that encode the same (in)dependencies that the learned DN. We have to point out that our proposal only changes the set of probability distributions but not the graph, so the model learned still has the same advantages about visualization. However, given that the relationships represented in a DN can be encoded by several BNs with different factorizations of the JPD, and that the conversion from DN to BN can not be attractive from the computational point of view, this proposal is based on a heuristic approach whose complexity order is linear in the number of dependencies found. The method proposed is shown in Figure 4.

```
1 For each variable X_i
2   For each Y_j parent of X_i
3     If X_i is also a parent of Y_j
4       If the conditioning set of X_i is
            grater that Y_j's
5         Y_j is removed as parent of X_i
6       Else
7         X_i is removed as parent of Y_j
```

Figure 4: Proposed method to obtain a more consistent set of CPDs.

We can call this new step in the learning process as parametric reduction. An important point in this procedure is in line 4. With this condition we want avoid large conditioning sets, what can reduce overfitting in the parameters estimation. The order in which the links can be traversed can be any although not all of then will yield the same solution. The reason for that is the heuristic nature of this algorithm and that a more sophisticated search would not be interesting for practical reasons. One of the benefits of DNs is ease of learning so we do not want to change that by introducing a complicated post-learning algorithm.

After performing this step is needed to recompute every probability distribution which has been modified. In the case that these probability distributions are in form of probability trees, if the removed variables are in the leaves the only thing to do is to aggregate its values to the up node in the tree, otherwise the entire tree should be re-built. However, if in the learing process we have cached the statistics the new tree can be built without computational cost. In the case of probability tables we can postpone leaning these tables after that step.

5 Experimental Results

This section is devoted to evaluate our proposal with some experiments. Our testing framework is base on the one used in (Heckerman et al., 2000) for testing probabilistic inference with real data. We use the same score function for a test dataset with N instances $\{d_1, \ldots, d_N\}$ and n variables:

$$score(d_1, \ldots, d_N | model) = -\frac{\sum_{i=1}^{N} \ln P(d_i|model)}{nN}. \quad (4)$$

However, instead of using real dataset we prefer using data sampled from known BNs. The reason is that we want focus only in parametrical learning and inference so if the real dependencies are known we can give this information to the different algorithms in order to avoid that the results were affected by the structural learning. Next we present a detailed description of our experimentation.

5.1 Description of the Experiments

We have selected seven BNs from different sources: `alarm` (Beinlich et al., 1989), `asia` (Lauritzen and Spiegelhalter, 1988), `car-starts` and `headache` (Elvira Consortium, 2002), `insurance` (Binder et al., 1992), `credit` (DSL) and `water` (Jensen et al., 1989), which is a dynamic network and we have use only the two first slices. Some details of these networks can be seen in Table 1. From each of these networks we have sampled two datasets with 5000 instances each one, one for training and one for testing.

We have defined eight models to make a comparison between them. First one is the reference model and is a BN in which the structure is fixed

Table 1: Set of Bayesian networks used in our experiments.

network	Num. vars	States range	Aver. states	MB range	Aver. MB
alarm	37	2-4	2.84	1-12	3.89
asia	8	2-2	2.00	1-5	2.50
car-starts	18	2-3	2.06	1-9	3.44
credit	12	2-4	2.83	2-6	3.67
headache	12	1-4	2.92	1-4	2.67
insurance	27	2-5	3.30	1-16	6.22
water	16	3-4	3.63	1-12	6.00

with the real links (BN-f). Second model is an empty network (Empty). Next we have three dependency networks models, one with probability tables in which links have been fixed from the real MB for each variable in the network (PT-f), other with probabilistic decision trees learned from data (PDT), and other with probabilistic decision trees but in which the search space for each PDT have been restricted to the real MB (PDT-f). In both cases we use the suggested value for $\kappa = 0.1$. For any of these three models we have another version in which we have used our method for reducing the CPDs. These new models are labeled with an asterisk (PT-f*, PDT* and PDT-f*). In all cases parameters are learned from data by using Laplace smoothing.

Every model has been learned with each training dataset. For all of them it has been computed their score (Equation (4)) with the test dataset. As the model BN-f is the reference one we have also obtained the absolute difference of score between each model and BN-f. This value is more informative because we are looking for models closer to the true probability distribution what is represented by BN-f. Besides, we have computed also the summation of all possible configurations, i.e. total joint probability, which should be equal to 1, but only for those models with a tractable number of configurations (asia, car-starts, credit, headache).

5.2 Results

In Table 2 we report the score value for every model and dataset. At the botton line we show the average value for each model. Lower values should indicate a better model, so all pure DN models should be taken as the best ones. However that does not make sense because they are even better than our reference model (BN-f) which represents the true JPD. The reason is that, as we have seen in Section 3, inconsistent DNs tend to have greater probability values in average so their score is lower. That is the reason why we prefer paying more attention to the difference with respect to the reference model.

Thus, these new results are shown in Table 3. There we can see that always the model closer to BN-f is the one in which we have applied our proposal. Specially the model based on probability tables is always the best one but in two datasets. Also is important to notice that our proposal improves the original model in every dataset for PT-f model. However, in PDT model our proposal deteriorates the accuracy in alarm and headache dataset, although in average its application improves the global accuracy.

Another interesting point is that PDT models without our proposal are much better than PT-f. That corroborate the idea that for DNs the use of more general encoding for the CPDs is advisable despite that this encoding is also an approximation in many cases.

Previous results give us an idea about the quality of those model. We can suppose that the increment in accuracy must be related with the reduction in the inconsistency. Additionally we have checked if the models encode a real probability distribution, i.e. whether the total joint probability for a given model is equal to one. This computation has been only done for the models learned with the smaller networks because this computation is computationally unfeasible for the others. The result is shown in Table 4. According to the table is clear that the pure DN models are quite far form being a probability distribution, but our proposal achieve that condition for all of them.

5.2.1 Time

Given that our proposal is a post-process to any learning algorithm, it seems that we will need more running time. In our experiments involving PDT we see that running time increases

Table 2: Score for each model and dataset.

	BN-f	Empty	PT-f	PT-f*	PDT	PDT*	PDT-f	PDT-f*
alarm	0.282	0.397	0.173	0.298	0.253	0.342	0.242	0.338
asia	0.287	0.342	0.224	0.289	0.225	0.287	0.225	0.289
car-starts	0.127	0.175	0.070	0.127	0.070	0.136	0.070	0.127
credit	0.879	0.959	0.765	0.886	0.807	0.888	0.807	0.900
headache	0.435	0.609	0.214	0.435	0.419	0.585	0.419	0.593
insurance	0.490	0.651	0.399	0.519	0.420	0.556	0.420	0.550
water	0.401	0.410	0.417	0.410	0.388	0.409	0.388	0.408
	0.414	0.506	0.323	0.423	0.369	0.458	0.367	0.458

Table 3: Absolute score difference between BN-f and the other models.

	Empty	PT-f	PT-f*	PDT	PDT*	PDT-f	PDT-f*
alarm	0.115	0.110	**0.015**	0.029	0.060	0.040	0.056
asia	0.055	0.062	0.002	0.062	**0.000**	0.062	0.002
car-starts	0.048	0.057	**0.000**	0.057	0.009	0.057	**0.000**
credit	0.080	0.114	**0.007**	0.071	0.009	0.071	0.021
headache	0.174	0.222	**0.000**	0.017	0.150	0.017	0.158
insurance	0.161	0.092	**0.029**	0.070	0.066	0.071	0.059
water	0.009	0.016	0.010	0.013	**0.008**	0.013	0.007
	0.092	0.096	**0.009**	0.046	0.043	0.047	0.043

Table 4: Total joint probability for tested models.

	BN-f	Empty	PT-f	PT-f*	PDT	PDT*	PDT-f	PDT-f*
asia	1.00	1.00	3.60	1.00	3.42	1.00	3.42	1.00
car-starts	1.00	1.00	20.04	1.08	11.40	1.00	11.40	1.00
credit	1.00	1.00	6.41	1.00	4.26	1.00	4.26	1.00
headache	1.00	1.00	29.68	1.00	5.70	1.00	5.70	1.00

1% in average, but also we have to take into account that we do not include the improvements explained at the end of Section 4. However with PT, if we assume that structural learning will be the same in both cases and compare our post-process and parameter learning we can see in Table 5 that we always obtain a faster algorithm. The reason is that our post-process makes the probability tables to estimate are much smaller and so that parametrical learning will be much faster because the complexity order is $O(M \times S)$ where M is the number of instances in the dataset and S is the size of each table.

Table 5: Percentage of run time our proposal can reduce the original algorithm.

dataset	% reduction
alarm	43
asia	13
car-starts	23
credit	18
headache	11
insurance	95
water	98

6 Conclusions

In this paper we have presented a method which aims to improve DNs. The main advantage of (general) DNs is that they can be learned from data easily, easier than BNs because of the lack of restrictions about cyclicity and easier than MNs because CPDs can be learn independently. Nonetheless this is also the main problem, the independent learning can lead to inconsistencies. They can be both structural and parametrical, however the later are more important. Whereas structural inconsistencies can be interesting for a better interpretation of the model (strong and weak dependencies), parametrical

ones deteriorate model performance.

Thus our proposal is based on improving DNs accuracy by reducing CPDs, because, as it has been seen in Section 3, the use of full distributions is the cause of that inconsistencies. Is worthy to point out that our proposal does not change the qualitative component of a model, i.e. its links. This new method, that can be seen as a post-learning stage, works by trying to recover a set of CPDs similar to a BN which represents the same relationships between variables. In order not to lose the computational advantage of the DNs learning we have chosen a heuristic approach which has a linear complexity order in the number of links. Its heuristic nature can be object of complaint, nonetheless in our experimentation we have made clear its benefit.

We plan to extend this work in two lines as future work. First we plan to make a deeper analysis of our proposal checking the performance with different sample sizes and different ordering in the reduction step and see whether it affects the results. Besides we want to test probabilistic queries for different set of variables with evidence in which we have to use Gibbs sampling. The second line of work is applying this method to scenarios where DNs have been use in order to improve their results, such as classifiers or Estimation of Distribution Algorithms.

Acknowledgments

This work has been partially supported by Spanish Ministerio de Educación y Ciencia (TIN2007-67418-C03-01); Junta de Comunidades de Castilla-La Mancha (PBI-08-048) and FEDER funds.

References

I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *2nd European Conf. on Artificial Intelligence in Medicine*, pages 247–256, 1989.

J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2):213–244, 1992.

W. Buntine. Theory refinement on bayesian networks. In *Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

Decision Systems Laboratory DSL. Genie. http://genie.sis.pitt.edu/.

Elvira Consortium. Elvira: An Environment for Creating and Using Probabilistic Graphical Models. In *1st European Workshop on Probabilistic Graphical Models*, pages 222–230, 2002. http://leo.ugr.es/elvira.

J. A. Gámez, J. L. Mateo, and J. M. Puerta. Dependency networks based classifiers: learning models by using independence test. In *3rd European Workshop on Probabilistic Graphical Models*, pages 115–122, 2006.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6: 147–156, 1984.

D. Heckerman, D. M. Chickering, and C. Meek. Dependency networks for inference, collaborative filtering and data visualization. *Machine Learning Research*, 1:49–75, 2000.

F. V. Jensen and T. D. Nielsen. *Bayesian networks and decision graphs*. Springer, 2007.

F. V. Jensen, U. Kjærulff, K. G. Olesen, and J. Pedersen. Et forprojekt til et ekspertsystem for drift af spildevandsrensning (an expert system for control of waste water treatment — a pilot project). Technical report, Judex Datasystemer A/S, Aalborg, Denmark, 1989.

S. L. Lauritzen. *Graphical Models*. Oxford Univesity Press, 1996.

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Royal Statistics Society, Series B*, 50: 157–194, 1988.

Sensitivity of Gaussian Bayesian networks to inaccuracies in their parameters

Miguel A. Gómez-Villegas and Paloma Main
Departamento de Estadística e Investigación Operativa
Universidad Complutense de Madrid
28040 Madrid, Spain

Rosario Susi
Departamento de Estadística e Investigación Operativa III
Universidad Complutense de Madrid
28040 Madrid, Spain

Abstract

To determine the effect of a set of inaccurate parameters in Gaussian Bayesian networks, it is necessary to study the sensitivity of the model. With this aim we propose a sensitivity analysis based on comparing two different models: the original model with the initial parameters assigned to the Gaussian Bayesian network and the perturbed model obtained after perturbing a set of inaccurate parameters with specific characteristics.

The network's outputs obtained for both models, after the evidence propagation, are going to be compared with the Kullback-Leibler divergence. This measure is useful to discriminate between two probability distributions, comparing the whole behavior of the considered probability distributions.

Depending on the set of parameters that are going to be perturbed, different expressions for the Kullback-Leibler are obtained. It is possible to determine the set of parameters that mostly disturb the network's output, detecting the variables that must be accurately described in the model.

The methodology developed in this work is for a Gaussian Bayesian network with a set of variables of interest and a set of evidential variables.

One example is introduced to show the sensitivity analysis proposed.

1 Introduction

In Bayesian networks some sensitivity analysis had been proposed to study the effect of inaccurate parameters over the network's output. Most of them, like the analyses and methodologies proposed by Laskey (1995), Coupé, van der Gaag and Habbema (2000), Kjærulff and van der Gaag (2000), Bednarski, Cholewa and Frid (2004) or Chan and Darwiche (2005), to name a few, are developed to study the sensitivity in discrete Bayesian networks. Some other papers discuss about general problems with different measures of sensitivity, like Pradhan, et al. (1996), Coupé and van der Gaag (2002) and Onisko and Druzdzel (2003).

In Gaussian Bayesian networks Castillo and Kjærulff (2003) performed a methodology based on studying small changes in the parameters, with one variable of interest in the model, and Gómez-Villegas, Main and Susi (2007) developed a sensitivity analysis to study any kind of perturbations, small or large changes in the parameters, when there exists one variable of interest in the Gaussian Bayesian network.
In the present work, we study a generalization of the sensitivity analysis proposed by

Gómez-Villegas, Main and Susi (2007), because now we consider a Gaussian Bayesian network with a set of variables of interest and a set of evidential variables. This approach is significatively different to the previous work because simultaneous perturbations in several parameters can be analyzed.

This paper is organized as follows. In Section 2 a brief introduction is presented, defining first a Bayesian network and a Gaussian Bayesian network and reviewing the evidence propagation for these models. Moreover, we introduce the working example. In Section 3, we present the methodology developed to study the sensitivity of a Gaussian Bayesian network with a set of variables of interest and in Section 4, we perform the sensitivity analysis proposed with the working example. Finally, the paper ends with some conclusions.

2 Gaussian Bayesian networks

A Bayesian network is a probabilistic graphical model useful to study a set of random variables with a specified dependence structure.
Bayesian networks have been studied by authors like Pearl (1988), Lauritzen (1996) or Jensen and Nielsen (2007), among others.

Definition 1 (Bayesian network)**.** A Bayesian network is a couple (G,P) where G is a directed acyclic graph (DAG) whose nodes are random variables $\mathbf{X} = \{X_1, ..., X_n\}$ and edges represent probabilistic dependencies, P=$\{p(x_1|pa(x_1)), ..., p(x_n|pa(x_n))\}$ being a set of conditional probability distributions (one for each variable), $pa(x_i)$ the set of parents of node X_i in G and $pa(x_i) \subseteq \{X_1, ..., X_{i-1}\}$.
The set P defines the joint probability distribution as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|pa(x_i)). \qquad (1)$$

Because of this modular structure, Bayesian networks are useful to study real life problems in complex domains.

Depending on the kind of variables of the problem, it is possible to describe discrete, Gaussian and mixed Bayesian networks. The results presented in this paper are developed for Gaussian Bayesian networks defined next

Definition 2 (Gaussian Bayesian network)**.** A Gaussian Bayesian network is a Bayesian network where the joint probability distribution of $\mathbf{X} = \{X_1, ..., X_n\}$ is a multivariate normal distribution $N(\mu, \Sigma)$, then the joint density

$$f(\mathbf{x}) = (2\pi)^{-n/2}|\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)\right\} \qquad (2)$$

where μ is the n-dimensional mean vector and Σ the $n \times n$ positive definite covariance matrix.

Moreover, the conditional probability distribution of X_i, satisfying expression (1), is a univariate normal distribution with density

$$f(x_i|pa(x_i)) \sim N(x_i|\mu_i + \sum_{j=1}^{i-1} \beta_{ij}(x_j - \mu_j), \nu_i)$$

where μ_i is the mean of the variables X_i, β_{ij} are the regression coefficients of X_i on its parents, 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 in the DAG. It should also be pointed that $pa(x_i) \subseteq \{X_1, ..., X_{i-1}\}$.

In Bayesian networks, when there exists evidence about one variable of the problem, knowing its value, the *evidence propagation* updates the probability distributions of the rest of the variables of the network given the evidence.
Different algorithms had been developed to propagate the evidence in Bayesian networks (see Jensen and Nielsen (2007)). In Gaussian Bayesian networks most of the algorithms proposed are based on computing the conditional probability distribution for a multivariate normal distribution given a set of evidential variables.
Thereby, to perform the evidence propagation in a Gaussian Bayesian network we consider a partition of the set of variables, where $\mathbf{X} = (\mathbf{E}, \mathbf{Y})'$, with $\mathbf{E}$ the set of evidential variables

and $\mathbf{Y}$ the rest of variables that will be considered as the set of variables of interest. After performing the evidence propagation, the conditional probability distribution of the variables of interest $\mathbf{Y}$ given the evidence $\mathbf{E} = \mathbf{e}$ is a multivariate normal distribution, $\mathbf{Y}|\mathbf{E} = \mathbf{e} \sim N(\mathbf{y}|\mu^{\mathbf{Y}|\mathbf{E}=\mathbf{e}}, \Sigma^{\mathbf{Y}|\mathbf{E}=\mathbf{e}})$ where

$$\mu^{\mathbf{Y}|\mathbf{E}=\mathbf{e}} = \mu_{\mathbf{Y}} + \Sigma_{\mathbf{YE}}\Sigma_{\mathbf{EE}}^{-1}(\mathbf{e} - \mu_{\mathbf{E}}) \quad (3)$$

and

$$\Sigma^{\mathbf{Y}|\mathbf{E}=\mathbf{e}} = \Sigma_{\mathbf{YY}} - \Sigma_{\mathbf{YE}}\Sigma_{\mathbf{EE}}^{-1}\Sigma_{\mathbf{EY}} \quad (4)$$

are the conditional mean vector and covariance matrix respectively.

Next, the working example of a Gaussian Bayesian network is introduced.

Example 1. The interest of the problem is about the duration of time that a machine works for. The machine is made up of 7 elements with random time to failure, X_i $i = 1, ..., 7$, connected as shown in the DAG of Figure 1.

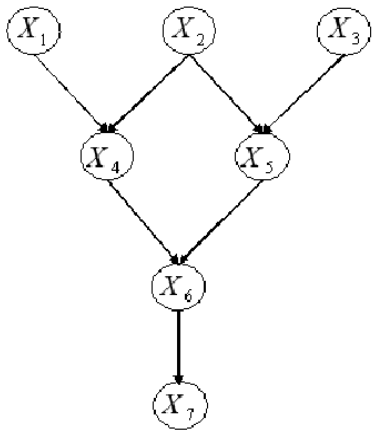


Figure 1: DAG of the Gaussian Bayesian network in Example 1

It is known that the time that each element is working is a normal distribution, being the joint probability distribution of $\mathbf{X} = \{X_1, X_2, ..., X_7\}$ a multivariate normal distribution $N(\mu, \Sigma)$ with parameters

$$\mu = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 1 \\ 4 \\ 5 \\ 8 \end{pmatrix}; \mathbf{\Sigma} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 2 & 2 \\ 0 & 1 & 0 & 2 & 2 & 8 & 8 \\ 0 & 0 & 2 & 0 & 2 & 4 & 4 \\ 1 & 2 & 0 & 6 & 4 & 20 & 20 \\ 0 & 2 & 2 & 4 & 10 & 28 & 28 \\ 2 & 8 & 4 & 20 & 28 & 97 & 97 \\ 2 & 8 & 4 & 20 & 28 & 97 & 99 \end{pmatrix}$$

The Gaussian Bayesian network that represents the problem is given by the joint probability distribution of $\mathbf{X} \sim N(\mathbf{x}|\mu, \Sigma)$ and by the DAG in Figure 1, showing the dependence structure between the variables of the example.

Experts know that the evidence is given by

$$\mathbf{E} = \{X_1 = 2, X_2 = 2, X_3 = 1\}$$

Then, after performing the evidence propagation, the probability distribution of the rest of the variables is $\mathbf{Y}|\mathbf{E} \sim N(\mathbf{y}|\mu^{\mathbf{Y}|\mathbf{E}}, \Sigma^{\mathbf{Y}|\mathbf{E}})$ with parameters

$$\mu^{\mathbf{Y}|\mathbf{E}} = \begin{pmatrix} 0 \\ 1 \\ -3 \\ 0 \end{pmatrix}; \mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}} = \begin{pmatrix} 1 & 0 & 2 & 2 \\ 0 & 4 & 8 & 8 \\ 2 & 8 & 21 & 21 \\ 2 & 8 & 21 & 23 \end{pmatrix}$$

The effect of introducing the evidence updates the parameters of the marginal distribution of the variables $\mathbf{Y}$ given by

$$\mu^{\mathbf{Y}} = \begin{pmatrix} 1 \\ 4 \\ 5 \\ 8 \end{pmatrix}; \mathbf{\Sigma}^{\mathbf{Y}} = \begin{pmatrix} 6 & 4 & 20 & 20 \\ 4 & 10 & 28 & 28 \\ 20 & 28 & 97 & 97 \\ 20 & 28 & 97 & 99 \end{pmatrix}$$

and the independence relationship because X_4 and X_5 become dependent.

3 Sensitivity Analysis

The aim of this work is to generalize the one way sensitivity analysis developed by

Gómez-Villegas, Main and Susi (2007) to a set of variables of interest.
The proposed methodology consists in comparing two different network's outputs: the first one, given by the network's output after the evidence propagation at the *original model*, and the other one, given by the network's output after the evidence propagation with a *perturbed model*. The perturbed model is obtained after adding a set of perturbations to the inaccurate parameters, as will be shown in Subsection 3.2. In this case, both network's outputs are the conditional probability distributions of the set of variables of interest, given the evidence.
It is useful to study the effect of inaccuracy over the parameters of a Gaussian Bayesian network for one variable of interest, after the evidence propagation. Nevertheless, now we can analyze simultaneous perturbations in several parameters.

3.1 Kullback-Leibler divergence

To compare the network's outputs we work with the n-dimensional Kullback-Leibler divergence (Kullback-Leibler, 1951). This measure takes into account the whole behavior of the distributions to be considered, therefore, it provides a suitable procedure to compare the network's outputs. The Kullback-Leibler (KL) divergence measure was introduced as a generalization of Shannon's entropy and has been used in statistical inference by authors like Jeffreys, Fisher and Lindley.

Definition 3 (Kullback-Leibler divergence)**.** Let $f(w)$ and $f'(w)$ be two probability densities defined over the same domain. The Kullback-Leibler divergence is given by

$$KL(f(w), f'(w)) = \int_{-\infty}^{\infty} f(w) \ln \frac{f(w)}{f'(w)} dw \quad (5)$$

When the probability densities to be compared with the KL divergence are multivariate normal distributions expression (5) can be written as

$$KL(f, f') = \\ = \frac{1}{2}\left[\ln \frac{|\mathbf{\Sigma}'|}{|\mathbf{\Sigma}|} + tr\left(\mathbf{\Sigma}\mathbf{\Sigma}'^{-1}\right) - \dim(\mathbf{X})\right] + \\ + \frac{1}{2}\left[(\mu' - \mu)^T \mathbf{\Sigma}'^{-1} (\mu' - \mu)\right] \quad (6)$$

where f is the joint probability density of $\mathbf{X} \sim N(\mathbf{x}|\mu, \mathbf{\Sigma})$ and f' is the joint probability density of $\mathbf{X} \sim N(\mathbf{x}|\mu', \mathbf{\Sigma}')$.

3.2 Sensitivity Analysis: methodology

The sensitivity analysis consists in comparing, with the KL divergence, two different network's output, obtained for the original and the perturbed model.
The original model is the initial description of the parameters of the network, given by $\mathbf{X} \sim N(\mathbf{x}|\mu, \Sigma)$. The perturbed model quantifies the uncertainty about the inaccurate parameters of the original model, as a set of additive perturbations. These are given by the *mean vector perturbations* δ and the *covariance matrix perturbations* Δ, where

$$\delta = \begin{pmatrix} \delta_{\mathbf{E}} \\ \delta_{\mathbf{Y}} \end{pmatrix}; \mathbf{\Delta} = \begin{pmatrix} \Delta_{\mathbf{EE}} & \Delta_{\mathbf{EY}} \\ \Delta_{\mathbf{YE}} & \Delta_{\mathbf{YY}} \end{pmatrix}$$

Depending on the inaccurate parameters it is possible to consider five different perturbed models obtained when the uncertainty is about the evidential means, the means of interest, the variances-covariances between evidential variables, the variances-covariances between variables of interest and about the covariances between evidential variables and variables of interest. Therefore, next perturbed models are considered:

- $\mathbf{X} \sim N(\mathbf{x}|\mu^{\delta_{\mathbf{E}}}, \Sigma)$ where

$$\mu^{\delta_{\mathbf{E}}} = \begin{pmatrix} \mu_{\mathbf{E}} + \delta_{\mathbf{E}} \\ \mu_{\mathbf{Y}} \end{pmatrix}$$

- $\mathbf{X} \sim N(\mathbf{x}|\mu^{\delta_{\mathbf{Y}}}, \Sigma)$ being

$$\mu^{\delta_{\mathbf{Y}}} = \begin{pmatrix} \mu_{\mathbf{E}} \\ \mu_{\mathbf{Y}} + \delta_{\mathbf{Y}} \end{pmatrix}$$

- $\mathbf{X} \sim N(\mathbf{x}|\mu, \Sigma^{\Delta_{\mathbf{EE}}})$ with

$$\mathbf{\Sigma^{\Delta_{EE}}} = \begin{pmatrix} \Sigma_{\mathbf{EE}} + \Delta_{\mathbf{EE}} & \Sigma_{\mathbf{EY}} \\ \Sigma_{\mathbf{YE}} & \Sigma_{\mathbf{YY}} \end{pmatrix}$$

- $\mathbf{X} \sim N(\mathbf{x}|\mu, \Sigma^{\Delta_{\mathbf{YY}}})$ where

$$\mathbf{\Sigma^{\Delta_{YY}}} = \begin{pmatrix} \Sigma_{\mathbf{EE}} & \Sigma_{\mathbf{EY}} \\ \Sigma_{\mathbf{YE}} & \Sigma_{\mathbf{YY}} + \Delta_{\mathbf{YY}} \end{pmatrix}$$

- $\mathbf{X} \sim N(\mathbf{x}|\mu, \Sigma^{\Delta_{\mathbf{YE}}})$ where

$$\mathbf{\Sigma^{\Delta_{YE}}} = \begin{pmatrix} \Sigma_{\mathbf{EE}} & \Sigma_{\mathbf{EY}} + \Delta_{\mathbf{EY}} \\ \Sigma_{\mathbf{YE}} + \Delta_{\mathbf{YE}} & \Sigma_{\mathbf{YY}} \end{pmatrix}$$

In this way, with the proposed sensitivity analysis the network's outputs of all the perturbed models are going to be compared with the network's output of the original model. Thereby, five different KL divergences are obtained, one for each perturbed model.
When the KL divergence is large for a specific perturbed model we can conclude that the set of parameters perturbed must be reviewed to describe the network more accurately. However, when the KL divergence is small, close to zero, it can be concluded that the network is not sensitive to the proposed perturbations. In summary, it can be established the set of inaccurate parameters that causes the worst perturbation. We have studied these cases separately to distinguish the effects of different kind of uncertain parameters. And we have also studied perturbations on parameters of both $\mathbf{Y}$ and $\mathbf{E}$, giving a robustness measure of the Gaussian Bayesian network (see Gómez-Villegas, Main and Susi (2008)).

3.3 Main results

The computations of the KL divergence for each perturbed model are in Propositions 1 and 2.

Proposition 1 (Uncertainty about the mean vector)**.** *Let (G, P) be a Gaussian Bayesian network with $\mathbf{X} = \{\mathbf{E}, \mathbf{Y}\}$ and $\mathbf{X} \sim N(\mathbf{x}|\mu, \Sigma)$ where the mean vector μ is uncertain. Giving values to the perturbations of the mean vector $\delta = (\delta_E, \delta_Y)^T$, the following results are obtained*

1. *When the perturbation $\delta_{\mathbf{E}}$ is added to the mean vector of the evidential variables, the perturbed model after the evidence propagation is $\mathbf{Y}|\mathbf{E}, \delta_{\mathbf{E}} \sim N(\mathbf{y}|\mu^{\mathbf{Y}|\mathbf{E},\delta_{\mathbf{E}}}, \mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}})$ with $\mu^{\mathbf{Y}|\mathbf{E},\delta_{\mathbf{E}}} = \mu^{\mathbf{Y}|\mathbf{E}} - \mathbf{\Sigma_{YE}}\mathbf{\Sigma_{EE}^{-1}}\delta_{\mathbf{E}}$. The KL divergence is*

$$KL^{\mu_{\mathbf{E}}} = \frac{1}{2}\left[\delta_{\mathbf{E}}^T M_1^T \left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\right)^{-1} M_1 \delta_{\mathbf{E}}\right]$$

   with $M_1 = \mathbf{\Sigma_{YE}}\mathbf{\Sigma_{EE}^{-1}}$

2. *When the perturbation $\delta_{\mathbf{Y}}$ is added to the mean vector of the variables of interest, after the evidence propagation the perturbed model is $\mathbf{Y}|\mathbf{E}, \delta_{\mathbf{Y}} \sim N(\mathbf{y}|\mu^{\mathbf{Y}|\mathbf{E},\delta_{\mathbf{Y}}}, \Sigma^{\mathbf{Y}|\mathbf{E}})$ where $\mu^{\mathbf{Y}|\mathbf{E},\delta_{\mathbf{Y}}} = \mu^{\mathbf{Y}|\mathbf{E}} + \delta_Y$ and the KL divergence is*

$$KL^{\mu_{\mathbf{Y}}} = \frac{1}{2}\left[\delta_{\mathbf{Y}}^T \left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\right)^{-1} \delta_{\mathbf{Y}}\right]$$

Proof. For uncertainty about the mean vector, we work with two perturbed models, depending on the set of inaccurate parameters.

In both perturbed models the covariance matrix $\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}$ is the same for the original model and for the perturbed model, then $tr\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\right)^{-1}\right) = \dim(\mathbf{Y})$. So, working with expression (6) and dealing with the perturbed models, the KL divergences follow directly.

□

Note also that the KL divergence obtained when there exists uncertainty about the mean vector of the evidential variables coincides with the KL divergence computed for a perturbation in the evidence vector $\mathbf{e}$. This gives us a tool to evaluate evidence influence on the network's outputs, as can be seen in Susi (2007).

Proposition 2 (Uncertainty about the covariance matrix)**.** *Let (G, P) be a Gaussian Bayesian network with $\mathbf{X} = \{\mathbf{E}, \mathbf{Y}\}$ and $\mathbf{X} \sim N(\mathbf{x}|\mu, \Sigma)$ where the covariance matrix Σ is uncertainty. Giving values to the perturbations of the covariance matrix $\mathbf{\Delta} = \begin{pmatrix} \Delta_{\mathbf{EE}} & \Delta_{\mathbf{EY}} \\ \Delta_{\mathbf{YE}} & \Delta_{\mathbf{YY}} \end{pmatrix}$, the following results are obtained*

1. *When the perturbation* $\mathbf{\Delta_{EE}}$ *is added to the variances-covariances of the evidential variables, after the evidence propagation, the perturbed model is*

$$\mathbf{Y}|\mathbf{E}, \mathbf{\Delta_{EE}} \sim N(\mathbf{y}|\mu^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{EE}}}, \mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{EE}}})$$

with $\mu^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{EE}}} = \mu_{\mathbf{Y}} + \mathbf{\Sigma_{YE}} \left(\mathbf{\Sigma_{EE}} + \mathbf{\Delta_{EE}}\right)^{-1} (\mathbf{e} - \mu_{\mathbf{E}})$ *and* $\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{EE}}} = \mathbf{\Sigma_{YY}} - \mathbf{\Sigma_{YE}} \left(\mathbf{\Sigma_{EE}} + \mathbf{\Delta_{EE}}\right)^{-1} \mathbf{\Sigma_{EY}}$

The KL divergence is

$$KL^{\mathbf{\Sigma_{EE}}} = = \frac{1}{2}\left[\ln \frac{\left|\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{EE}}}\right|}{\left|\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\right|} - \dim(\mathbf{Y})\right] + \frac{1}{2}\left[tr\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{EE}}}\right)^{-1}\right)\right] + \frac{1}{2}\left[M_2^T\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{EE}}}\right)^{-1} M_2\right]$$

where $M_2 = \mu^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{EE}}} - \mu^{\mathbf{Y}|\mathbf{E}}$.

2. *When the perturbation* $\mathbf{\Delta_{YY}}$ *is added to the variances-covariances between the variables of interest, after the evidence propagation the perturbed model is*

$$\mathbf{Y}|\mathbf{E}, \mathbf{\Delta_{YY}} \sim N(\mathbf{y}|\mu^{\mathbf{Y}|\mathbf{E}}, \mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{YY}}})$$

with $\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{YY}}} = \mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}} + \mathbf{\Delta_{YY}}$.

The obtained KL divergence is

$$KL^{\mathbf{\Sigma_{YY}}} = = \frac{1}{2}\left[\ln \frac{\left|\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}} + \mathbf{\Delta_{YY}}\right|}{\left|\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\right|} - \dim(\mathbf{Y})\right] + \frac{1}{2}\left[tr\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}} + \mathbf{\Delta_{YY}}\right)^{-1}\right)\right]$$

3. *If the perturbation* $\mathbf{\Delta_{YE}}$ *is added to the covariances between* $\mathbf{Y}$ *and* $\mathbf{E}$*, the perturbed model after the evidence propagation is*

$$\mathbf{Y}|\mathbf{E}, \mathbf{\Delta_{YE}} \sim N(\mathbf{y}|\mu^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{YE}}}, \mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{YE}}})$$

with $\mu^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{YE}}} = \mu_{\mathbf{Y}} + (\mathbf{\Sigma_{YE}} + \mathbf{\Delta_{YE}})\,\mathbf{\Sigma_{EE}^{-1}}(\mathbf{e} - \mu_{\mathbf{E}})$ *and* $\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E},\mathbf{\Delta_{YE}}} = \mathbf{\Sigma_{YY}} - (\mathbf{\Sigma_{YE}} + \mathbf{\Delta_{YE}})\,\mathbf{\Sigma_{EE}^{-1}}\,(\mathbf{\Sigma_{EY}} + \mathbf{\Delta_{EY}})$

Then, the KL divergence is

$$KL^{\mathbf{\Sigma_{YE}}} = = \frac{1}{2}\left[\ln \frac{\left|\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}} - M(\mathbf{\Delta_{YE}})\right|}{\left|\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\right|} - \dim(\mathbf{Y})\right] + \frac{1}{2}\left[tr\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}}\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}} - M(\mathbf{\Delta_{YE}})\right)^{-1}\right)\right] + \frac{1}{2}\left[(\mathbf{e} - \mu_{\mathbf{E}})^T\left(\mathbf{\Sigma_{EE}^{-1}}\right)^T M_3 \mathbf{\Sigma_{EE}^{-1}}(\mathbf{e} - \mu_{\mathbf{E}})\right]$$

where

$$M_3 = \mathbf{\Delta_{YE}^T}\left(\mathbf{\Sigma}^{\mathbf{Y}|\mathbf{E}} - \mathbf{\Delta_{YE}}\mathbf{\Sigma_{EE}^{-1}}\mathbf{\Sigma_{YE}^T} - \mathbf{\Sigma_{YE}}\mathbf{\Sigma_{EE}^{-1}}\mathbf{\Delta_{EY}} - \mathbf{\Delta_{YE}}\mathbf{\Sigma_{EE}^{-1}}\mathbf{\Delta_{EY}}\right)^{-1}\mathbf{\Delta_{YE}}$$

Proof. We work with three perturbed models defined for different sets of inaccurate parameters. The corresponding conditional parameters for the perturbed model are stated. Then, computing the KL divergence with (6) to compare the network's output of the original model with the network's outputs obtained for the perturbed models, the obtained expressions follow directly. □

Although computation involves matrix operations and depends on the network size, the final calculations consider a reduced dimension because of the partition in the original covariance matrix. The introduced results can be implemented algorithmically with a polynomial computational complexity.

4 Experimental results

Next, we will run the sensitivity analysis proposed in Section 3 for the Example 1.

Example 2. There are different opinions between experts about the parameters of the Gaussian Bayesian network shown in

Example 1. Quantifying this uncertainty we obtain the perturbed mean vector δ and the perturbed covariance matrix Δ as follows

$$\delta_{\mathbf{E}} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}; \delta_{\mathbf{Y}} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix}$$

$$\mathbf{\Sigma_{EE}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$\mathbf{\Sigma_{YY}} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 \\ 1 & 0 & -2 & 0 \end{pmatrix}$$

$$\mathbf{\Sigma_{EY}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Taking into account the evidence $\mathbf{E} = \{X_1 = 2, X_2 = 2, X_3 = 1\}$ and the variables of interest $\mathbf{Y} = \{X_4, X_5, X_6, X_7\}$, it is possible to perform the sensitivity analysis proposed.
Then, for the KL divergence with the expressions presented in Propositions 1 and 2, next values are obtained:

$$KL^{\mu_{\mathbf{E}}} = 2.125$$
$$KL^{\mu_{\mathbf{Y}}} = 2.375$$
$$KL^{\mathbf{\Sigma_{EE}}} = 0.596$$
$$KL^{\mathbf{\Sigma_{YY}}}(f, f^{\mathbf{\Sigma_{YY}}}) = 1.629$$
$$KL^{\mathbf{\Sigma_{YE}}}(f, f^{\mathbf{\Sigma_{YE}}}) = 0.265$$

With the obtained results it is possible to conclude that some parameters must be reviewed to describe the network more accurately. The parameters that must be reviewed are the mean vector, because the possible perturbations makes the KL divergence larger than 1 and, moreover, it is necessary to review the parameters that describe the variances-covariances between the variables of interest because the network is sensitive to uncertainty about these parameters.
Uncertainty about the variances-covariances between evidential variables and about the covariances between variables of interest and evidential variables do not change the network's output so much, therefore the network is not sensitive to these inaccurate parameters.

5 Conclusions

In a Gaussian Bayesian network, some inaccuracies about the parameters that describe the network, involve a sensitivity analysis of the model. In this paper we propose a sensitivity analysis for Gaussian Bayesian networks, useful to determine the set or sets of inaccurate parameters that must be reviewed to be introduced in the network more accurately, or if the network is not sensitive to inaccuracies.
The analysis performed is a generalization of the one way sensitivity analysis developed by Gómez-Villegas, Main and Susi (2007), working now with a set of variables of interest and being able to analyze a set of parameters perturbations simultaneously.
At the proposed sensitivity analysis five different sets of parameters are considered, depending on the type of variables and if they describe the mean or the covariance of the model. After computing the expressions of the KL divergence obtained in Propositions 1 and 2, it is possible to conclude the set or sets of parameters that must be reviewed to describe the network more properly. In this way, when a KL divergence is small, next to zero, we can conclude that the network is not sensitive to the proposed perturbations.
The methodology we present is easy to perform with any Gaussian Bayesian network and is useful to evaluate any kind of inaccurate parameters, that is, large and small perturbations associated to uncertain parameters.

Acknowledgments

This research was supported by the Ministerio de Educación y Ciencia from Spain Grant MTM2005–05462 and Comunidad de Madrid–Universidad Complutense Grant

References

Bednarski, M., Cholewa, W. and Frid, W. 2004. Identification of sensitivities in Bayesian networks. *Engineering Applications of Artificial Intelligence*, 17:327–335.

Castillo, E. and Kjærulff, U. 2003. Sensitivity analysis in Gaussian Bayesian networks using a symbolic-numerical technique.*Reliability Engineering and System Safety*, 79:139–148.

Chan, H. and 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. and Habbema, J.D.F. 2000. Sensitivity analysis: an aid for belief-network quantification.*The Knowledge Engineering Review*, 15(3):215–232.

Coupé, V.M.H. and 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., Main, P. and Susi, R. 2007. Sensitivity Analysis in Gaussian Bayesian Networks Using a Divergence Measure. *Communications in Statistics: Theory and Methods*, 36(3):523–539.

Gómez-Villegas, M.A., Main, P. and Susi, R. 2008. The effect of block parameter perturbations in Gaussian Bayesian networks: Sensitivity and Robustness. (Submitted).

Jensen, F.V. and Nielsen, T.D. 2007.*Bayesian Networks and Decision Graphs*. New York, Springer Verlag.

Kjærulff, U. and van der Gaag, L.C. 2000. Making Sensitivity Analysis Computationally Efficient. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, pages 315–325. Morgan Kaufmann.

Kullback, S. and Leibler, R.A. 1951. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22:79–86.

Laskey, KB. 1995. Sensitivity Analysis for Probability Assessments in Bayesian Networks. *IEEE Transactions on Systems, Man and Cybernetics*, 25:901–909.

Lauritzen, S.L. 1996. *Graphical Models*. Oxford, Clarendon Press.

Onisko, A. and Druzdzel, M.J. 2003. Effect of Imprecision in Probabilities on Bayesian Network Models: An Empirical Study *Working notes of the European Conference on Artificial Intelligence in Medicine*.

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

Pradhan, M., Henrion, M., Provan, G., del Favero, B. and Huang, K. 1996. The Sensitivity of Belief Networks to Imprecise Probabilities: An Experimental Investigation. *Artificial Intelligence*, 85(1–2):363–397.

Susi R. 2007. *Análisis de Sensibilidad en Redes Bayesianas Gaussianas*. Ph.D. Thesis, Departamento de Estadística e Investigación Operativa, Universidad Complutense de Madrid, Spain.

Approximate representation of optimal strategies from influence diagrams

Finn Verner Jensen
Department of Computer Science
Aalborg University
9220 Aalborg, Denmark
fvj@cs.aau.dk

Abstract

There are three phases in the life of a decision problem, specification, solution, and representation of solution. The specification and solution phases are off-line, while the represention of solution often shall serve an on-line situation with rather tough constraints on time and space. One of the advantages of influence diagrams (IDs) is that for small decision problems, the distinction between phases does not confront the decision maker with a problem; when the problem has been properly specified, the solution algorithms are so efficient that the ID can also be used as an on-line representation of the solution. If the solution algorithm cannot meet the on-line requirements, you will construct an alternative structure for representing the optimal strategy, for example a look-up table or a strategy tree. We report on ongoing work with situations where the solution algorithm is too space and time consuming, and where the policy functions for the decisions have so large domains that they cannot be represented directly in a strategy tree. The approach is to have separate ID representations for each decision variable. In each representation the actual information is fully exploited, however the representation of policies for future decisions are approximations. We call the approximation *information abstraction*. It consists in introducing a dummy structure connecting the past with the decision. We study how to specify, implement and learn information abstraction.

1 Introduction

There are several algorithms for solving IDs (Shachter, 1986),(Shenoy, 1992), (Jensen et al., 1994), but the principle behind them all is dynamic programming starting with the last decision. That is, first an optimal policy for the last decision is determined. Next, this policy is represented somehow, and the optimal policy for the second last decision is determined by using the policy for the last decision for forecasting the future. To illustrate this process, consider the ID in Figure 1. We can represent the optimal policy for the last decision as a conditional probability table (CPT), and the ID from Figure 1 is transformed to the one in Figure 2.

When the solution process is over, you may represent the optimal strategy as a set of CPTs representing the policies, and you will have the Bayesian network in Figure 3.

This process is off-line, and it may require very much space and time. However, as long as the task is tractable the solution phase is not really a problem.

Note that if the ID contains only one decision, then the information variables are instantiated when a decision is to be taken, and the task is reduced to propagation in a Bayesian network. That is, you need not specify the parents of the decision variable, and you have a very compact and efficient representation of the optimal policy.

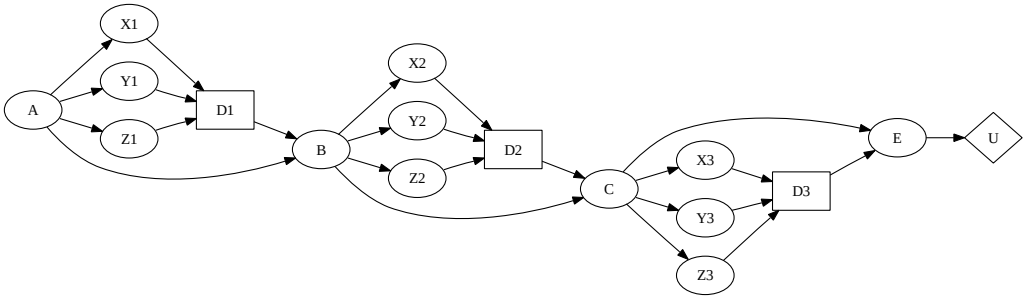


Figure 1: We shall refer to this ID throughout as an illustrating example.

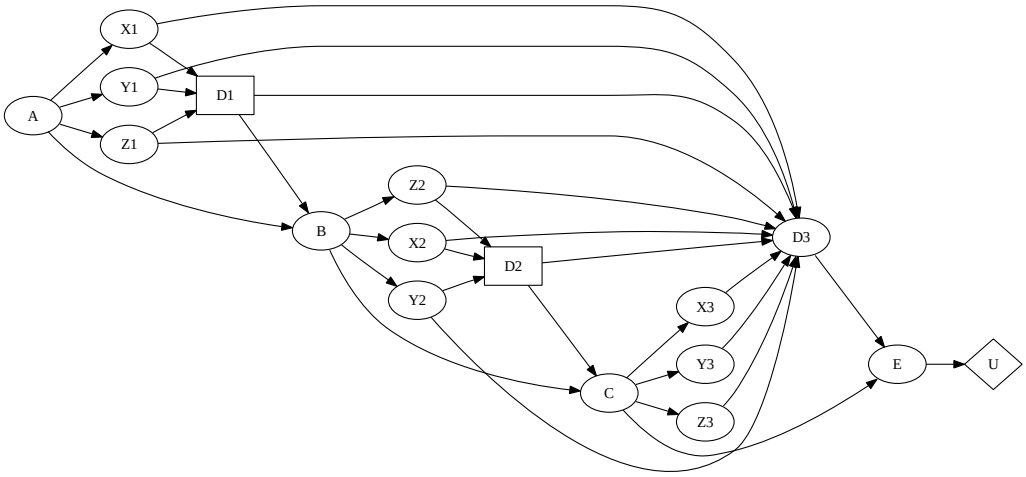


Figure 2: The node for the last decision is substituted with a chance node representation of a policy.

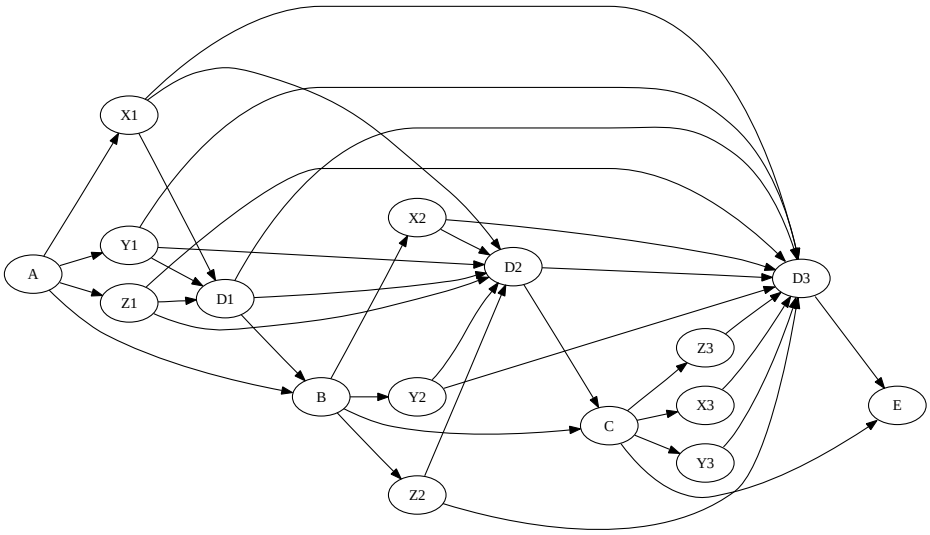


Figure 3: A representation of a strategy. Actually, if we are not interested in the expected utilities, we only need the conditional probabilities for the D-nodes

2 On-line use of a solved influence diagram

We look at situations where the influence diagram is too complex for on-line use. In particular, we consider the situation where the domains for the policies are too large. We shall assume that this is the case for the ID in Figure 1.

Now, consider the last decision, $D3$. When you decide on $D3$, you know the states of $X1, Y1, Z1, D1, X2, Y2, Z2, D2, X3, Y3$, and $Z3$. Then the model in Figure 4 can be used. The information is entered as evidence, and the expected utilities for $D3$ are easily calculated.

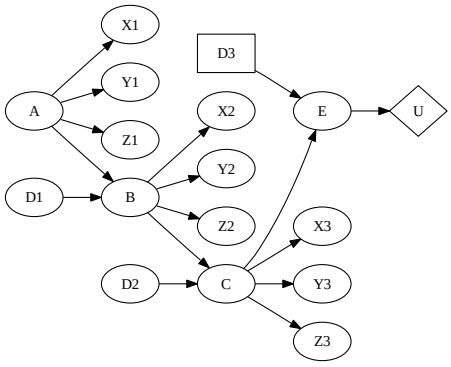


Figure 4: An ID for on-line representation of the last decision.

On the other hand, when deciding $D2$, you need to calculate the expected utility of each option, and in order to do that you need to know the policy δ_{D3} for the future decision. Representing δ_{D3} as in Figure 2 is intractable. You need an alternative representation, and you have to settle with an approximate prepresentation. What is crucial for an approximate representation is that it for each configuration over $past(D2)$ reflects the order of expected utilities for $D2$. (If X is a decision variable, then $past(X)$ denotes the set of variables known at the time of deciding on X).

3 Information abstraction

An approximation approach is *information abstraction*: introduce extra structure connecting the information with the decision. There are several schemes for information abstraction. For example, $past(D2)$ way be represented by a *history variable*. An immediate representation would be a chance node H with $past(D2)$ as parents, but if you were faced with intractably learge policy domains, then the CPT for H will also be intractable. Another representation can be a *history belt* (see Figure 5).

Domain knowledge can help to determine a good way of introducing history variables. We shall later discuss ways of learning history variables from the initial ID specification.

We propose another scheme for information abstraction, *conditional decomposition of domains*. Let δ_D be a policy for a decision variable D. A way of decomposing the domain of δ_D would be to assume that the policy has the form:

if f(X) **then** g(Y) **else** h(Z),

where X, Y, Z are subsets of the domain, and f is a Boolean valued function. The function f may be an alert function. For example, an unmanned vehicle will focus on fullfilling its mission unless an alert tells it to return for more fuel. In a game scenario, you may try to meet your own goals. However, if your opponent is close to achieving hers, you must focus on blocking her way.

If you from domain knowledge know how δ_D can be decomposed, you can exploite it in the solution phase, and you may not need an approximate representation after all. Using the language *sequential influence diagrams* (SIDs) (Jensen et al, 2006), a specification would look like the one in Figure 6, where $X = \{E, F\}, Y = \{C, E\}, Z = \{F, G\}$.

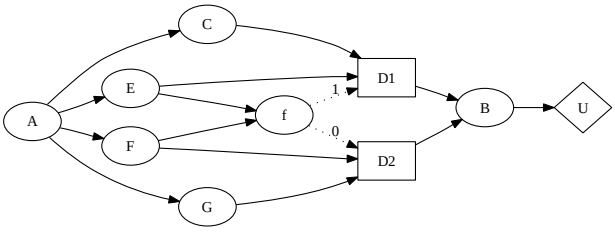


Figure 6: An SID representing a decision with a policy of the form **if** $f(E, F)$ **then** $g(C, E)$ **else** $h(E, G)$.

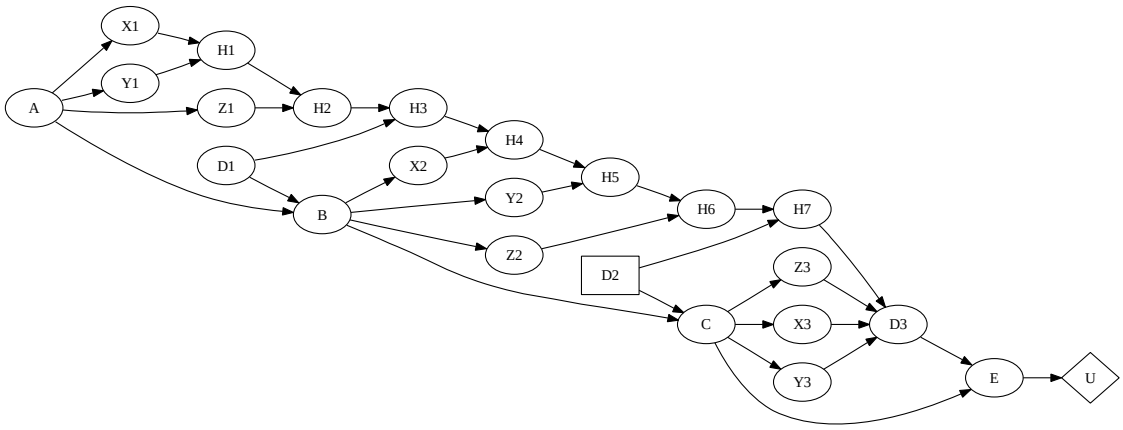


Figure 5: An ID for representing δ_{D2}. δ_{D3} is approximated through a belt of history variables ending with $H7$, wich is assumed to be observed.

The specification in Figure 6 can be translated to two IDs; one for determining $\delta_{D1}(C, E)$ with $f = 1$ inserted as evidence, and one for δ_{D2} with $f = 0$ inserted.

In an ID with several decisions, the decomposed policy from Figure 6 can be represented as in Figure 7.

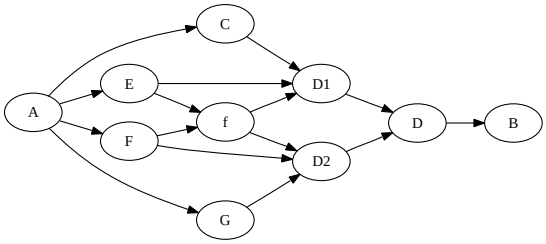


Figure 7: A BN representation of the decomposed policy from Figure 6. The nodes $D1$ and $D2$ have an extra state "no decision".

As indicated above, conditional decomposition may come up naturally from the domain. We will later discuss how artificial decompositions may be learned from the ID specification.

3.1 Overestimation of information

The abstractions presented above underestimate the information available when the decision actually is taken. The information is used to estimate the distribution of the parents of the relevant utility functions, and with abstraction of information, this estimate will be less precise.

You may also overestimate the information. For example, consider the ID in Figure 8.

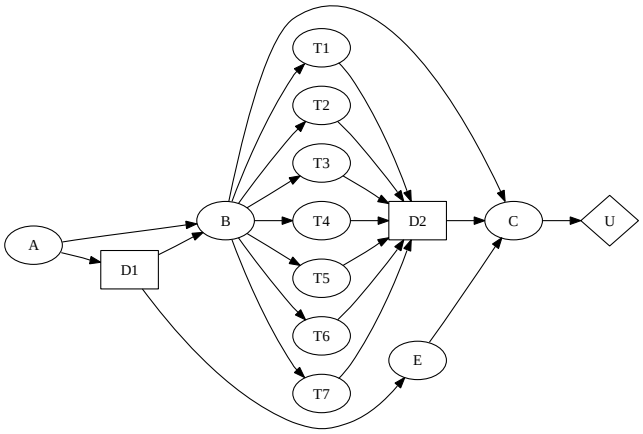


Figure 8: An ID with seven different tests for the same variable.

When deciding $D1$, it may be reasonable to say that when you in the future decide $D2$ you will know the state of B, and a good approximating ID for calcultating the policy for $D1$ will be the one in Figure 9.

We shall not pursue overestimation of information further in this paper.

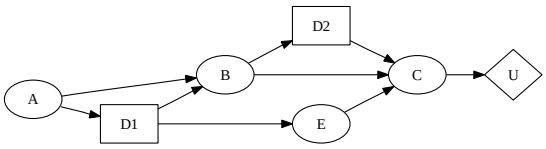


Figure 9: An ID for calculating a policy for $D1$ in Figure 8.

4 Space issues

Just to get an impression of the space issues, consider the example ID in Figure 1. Let the decision nodes and the observed nodes have 5 states, and let the background nodes A, B, C and E have 25 states. Then the maximal clique in a minimal clique junction tree is of size $6,3 \cdot 10^9$. The maximal clique size in a junction tree for the representation in Figure 3 is $2,5 \cdot 10^8$.

If we in the representation with history variables (Figure 1) let $H1$ have ten states and increase the number of states by two up to $H7$ with 22 states, then a junction tree for the model in Figure 1 has size $3,5 \cdot 10^6$, and the maximal clique has size $1,5 \cdot 10^6$. If we instead start with six states for $H1$, increase by one up to $H7$ with 12 states then the junction tree size is 430,000 and the maximal clique has size 187,000.

If we approximate δ_{D3} with the policy "**if** $f(Z1, Z2, Z3)$ **then** $g(X1, X2, X3, D1, D2)$ **else** $h(Y1, Y2, Y3, D1, D2)$", then the junction tree has maximal clique size of $3,9 \cdot 10^6$ and if we use "**if** $f(Z1, Z2, Z3)$ **then** $g(X1, X2, X3)$ **else** $h(Y1, Y2, Y3)$", then the maximal clique size is 165,000.

5 Learning information abstraction

Consider Figure 4, and assume that we wish to learn a representation of δ_{D3} as used in Figure 5. In order to do so you can establish a sample by exploiting a representation proposed by (Cooper, 1988). The decision node is substituted by a chance node with even priors, and the utility node is substituded by a chance node with two states (1 and 0). The conditional probability table $P(U = 1|pa(U))$ represents normalized utilities. For the resulting network it holds that $P(D|U = 1, e)$ is proportional to the expected ulitities for D given the evidence e. Now, sample from the network with $U = 1$ inserted. By removing the unobserved variables (and U) from the table, you have a sample representing $P(D|pa(D))$, which is proportional to the expected utilities of D given $pa(D)$.

5.1 An example

Take the simple example in Figure 10. The parameters in the model are so that the policy for D is characterized by the three functions $(f(Z), g(X), h(Y))$ in the following way:

if $Z = y$ **then**
(**if** $X = y$ **then** $D = a_1$ **else** $D = a_2$)
else (**if** $Y = y$ **then** $D = a_3$ **else** $D = a_4$).

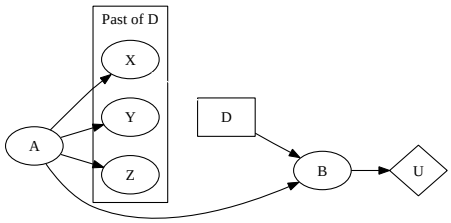


Figure 10: A simple ID to illustrate learning of information abstraction.

The model is transformed to the Bayesian network in Figure 11, where $P(D|X, Y, Z, U = 1)$ is proportional to $EU(D|X, Y, D)$, and therefore $\delta_D(X, Y, Z) = argmax_D P(D|X, Y, Z, U = 1)$.

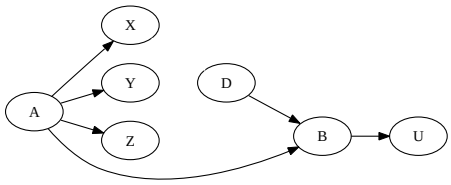


Figure 11: The Bayesian network for sampling. $U = 1$ is inserted before sampling

We sampled 10.000 cases from the model in Figure 11 with $U = 1$ inserted, and we used the EM algorithm (Lauritzen, 1995) to learn the unknown parameters $P(f|Z), P(DX|f, X)$, and $P(DY|f, Y)$ in Figure 12.

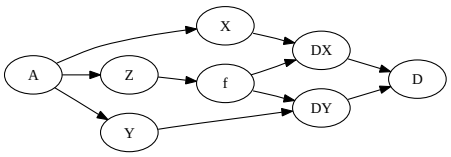


Figure 12: A decomposition model. The CPTs for f, DX, and DY are unknown

The CPT for D reflects that if $f = 1$ then $D = DX$, and if $f = 0$ then $D = DY$. The learning resulted in CPTs for (f, DX, and DY) which are very close to the functions (f, g, h). Finally, we modify the CPTs to give probability 1 to the state of maximal probability, and we ended up with the correct policy.

If you do not know the form of the decomposition, then you have to experiment with different structures. For this example, we tried to learn a policy characterized by the three functions $(f(X), g(Z), h(Y))$. The resulting structure had 5 out of 8 decisions correct.

In the model in Figure 13 we have introduced the history variables $H1$ with three states, and $H2$ with four states. The learning procedure resulted in parameters such that the correct decision in all eight cases have maximal probability. To modify the tables to give probability 1 to the decision with maximal probability, you can use various tuning methods (see for example (Jensen and Nielsen, 2006)).

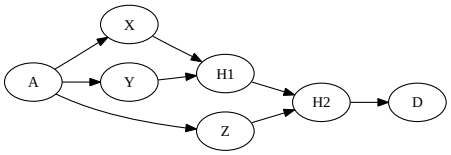


Figure 13: A model with history variables. The CPTs for $H1, H2$, and D are unknown.

6 Discussion and future work

The size of domains of decision policies is a major obstacle in practical use of decision theory for decision problems involving a sequence of decisions. We have in this paper analysed the problem and we have proposed some schemes for addressing the problem. First of all we need a larger set of schemes for information abstraction, and we need experience with underestimation as well as overestimation of information. Issues to study will be complexity as well as precision. We have in this paper presented a method for learning parameters when the structure for information abstraction is given. Learning structure is more intricate. As neither the number of latent variables nor the number of states of the variables are known, the only method known to us is trial and error. We need a systematic way of passing through possible structures.

An alternative method for addressing intractably large decision domains is LIMIDs (Nilsson and Lauritzen, 2001). The approach is to remove some variables from the domains. For the ID in Figure 1, a LIMID structure could be that the decision maker only knows the current information and the previous decision. This is illustrated in Figure 14, where the policies are represented as CPTs for $D1, D2$ and $D3$.

Nilsson and Lauritzen propose an iterative procedure for determining approximate optimal policies: start with an arbitrary set of policies and solve the three single decision IDs; use the calculated policies as CPTs and solve the new single decision ID. Continue so untill no policy is changed.

Following the approach presented in this paper, we would solve the last decision through sampling and use of the EM algorithm to determine a policy for $D3$; then use this policy to determine a policy for $D2$ through sampling and EM, and eventually solve the ID for $D1$. It is an interesting issue for further research to compare these two approaches.

References

Gregory F. Cooper (1988). A method for using belief networks as influence diagrams. *Fourth Workshop on Uncertainty in Artificial Intelligence*: 55–63.

Finn V. Jensen, Thomas D. Nielsen and Prakash P Shenoy (2006). Sequential influence diagrams: A unified asymmetry framework. *International Journal of Approximate Reasoning*, 42(1–2): 101-118.

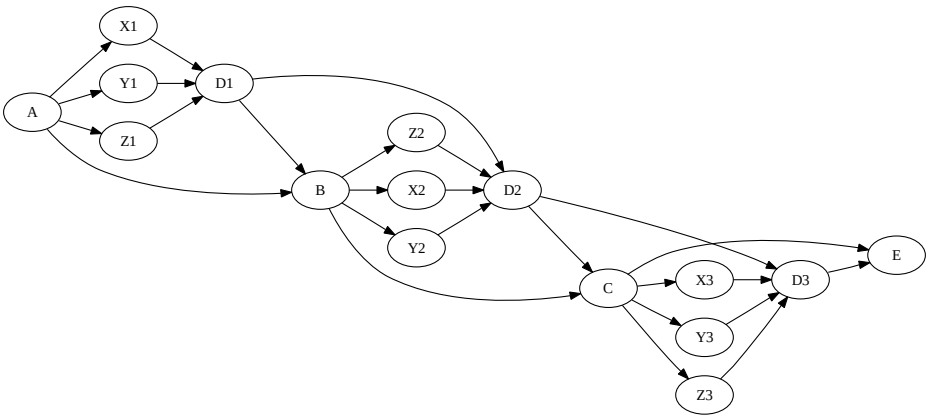


Figure 14: A LIMID structure representing decisions where only the current information and the previous decision are included in the decision domain

Finn V. Jensen, Thomas D. Nielsen (2007). *Bayesian Networks and Decision Graphs. Springer, New York.*

Frank Jensen, Finn V. Jensen and Søren L. Dittmer (1994). From Influence Diagrams to Junction trees. *Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann: 367–374.

Steffen L. Lauritzen (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19: 191–201.

Dennis Nilsson and Steffen L. Lauritzen (2001). Representing and solving decision problems with limited information. *Management Science*, 47: 1235-1251.

Ross Shachter (1986). Evaluating influence diagrams. *Operations Research*, 34(6): 871–882.

Prakash P. Shenoy (1992). Valuation Based Systems for Bayesian Decision Analysis. *Operations Research*, 40(3): 463–484.

Complexity Results for Enumerating MPE and Partial MAP

Johan Kwisthout
Department of Information and Computer Sciences, Utrecht University
P.O. Box 80.089, 3508TB Utrecht, The Netherlands
johank@cs.uu.nl

Abstract

While the computational complexity of finding the most likely joint value assignment given full (MPE) or partial (Partial MAP) evidence is known, less attention has been given to the related problem of finding the k-th most likely assignment, for arbitrary values of k. Yet this problem has very relevant practical usages, for example when we are interested in a list of alternative explanations in decreasing likeliness. In this paper a hardness proof of enumerating Most Probable Explanations (MPEs) and Maximum A-Priori Probabilities (Partial MAPs) is given. We prove that finding the k-th MPE is P^{PP}-complete, and prove that finding the k-th Partial MAP is $\mathsf{P}^{\mathsf{PP}^{\mathsf{PP}}}$-complete.

1 Introduction

An important problem that rises from the practical usage of probabilistic networks (Jensen, 2007; Pearl, 1988) is the problem of finding the most likely value assignment to a set of variables, given full or partial evidence. When the evidence is equal to the entire complement of that set in the network, the problem is known as the Most Probable Explanation or MPE-problem[1]. Finding, or even approximating, such a value assignment is NP-hard (Shimony, 1994; Bodlaender et al., 2002; Abdelbar and Hedetniemi, 1998). On the other hand, finding the most likely value assignment, given evidence for a *subset* of the complement set (the Partial MAP-problem), is even harder: Park and Darwich proved (2004) that this problem is $\mathsf{NP}^{\mathsf{PP}}$-complete and remains NP-complete on polytrees.

In practical applications, one often wants to find a number of different value assignments with a high likeliness, rather than only the most likely assignment (see e.g. Santos Jr. (1991) or Charniak and Shimony (1994)). For example, in medical applications one wants to suggest alternative (but also likely) explanations to a set of observations. One might like to prescribe medication that covers a number of plausible causes, rather than only the most probable cause. It may be useful to examine the second-best explanation to gain insight in *how good* the best explanation is, relative to other solutions, or, how sensitive it is to changes in the parameters of the network (Chan and Darwiche, 2006).

While algorithms exist that can sometimes find k-th best explanations fast, once the best explanation is known (Charniak and Shimony, 1994), it has been shown that calculating or even approximating the k-th best explanation is NP-hard (Abdelbar and Hedetniemi, 1998), whether the best explanation is known or not. Nevertheless, the exact complexity of this problem has not been established yet.

The complexity of finding k-th best assignments to the Partial MAP-problem has, to our best knowledge, not yet been investigated. However, in many applications it is unlikely that full evidence of the complement of the variables of interest in the network is available. For example, in the *Oesophagus Network*, a probabilistic network for patient-specific therapy selection for oesophageal cancer (van der Gaag et al., 2002), a number of variables (like the

[1] In the literature also denoted as Maximum Probability Assignment (MPA) or Maximum A-posteriori Probability (MAP).

presence of haematogenous metastases or the extent of lymph node metastases) are intermediate, non-observable variables. Likewise, the ALARM network (Beinlich et al., 1989) has sixteen observable and thirteen intermediate variables. Therefore, the problem of finding k-th best assignments, given *partial* evidence, may be even more relevant in practical applications than the corresponding problem where full evidence is available.

In this paper, we extend the problem of finding the most likely value assignment to the problem of enumerating joint value assignments, i.e., finding the k-th likely assignment for arbitrary values of k, with either full or partial evidence. We will prove that (decision variants of) these problems are complete for the complexity classes P^{PP} and $\mathsf{P}^{\mathsf{PP}^{\mathsf{PP}}}$, respectively, suggesting that these problems are much harder than the (already intractable) restricted cases where $k = 1$, and also much harder than the PP-complete INFERENCE problem. Furthermore, while some problems are known to be P^{PP}-complete, finding the k-th Partial MAP is (to our best knowledge) the first problem with a practical application that is shown to be $\mathsf{P}^{\mathsf{PP}^{\mathsf{PP}}}$-complete, making this problem interesting from a more theoretical viewpoint as well.

This paper is organized as follows. First, in Section 2, we will briefly introduce probabilistic networks and introduce a number of concepts from computational complexity theory. We will discuss the complexity of enumerating value assignment with full, respectively partial, evidence in Sections 3 and 4. In Section 5 we conclude this paper.

2 Preliminaries

A probabilistic network $\mathbf{B} = (\mathbf{G}, \Gamma)$ is defined by a directed acyclic graph $\mathbf{G} = (\mathbf{V}, \mathbf{A})$, where $\mathbf{V} = \{V_1, \ldots, V_n\}$ models a set of stochastic variables and $\mathbf{A}$ models the (in)dependences between them, and a set of parameter probabilities Γ, capturing the strengths of the relationships between the variables. The network models a joint probability distribution $\Pr(\mathbf{V}) = \prod_{i=1}^{n} \Pr(v_i \mid \pi(V_i))$ over its variables. We will use bold upper case letters to denote sets of variables (i.e., subsets of $\mathbf{V}$) and bold lower case letters to denote particular value assignments to these sets. The set of observed variables (the *evidence* variables) will be denoted as $\mathbf{E}$, and the observations themselves as $\mathbf{e}$. We will use $\Pr(\mathbf{v} \,|\, \mathbf{e})$ as a shorthand for $\Pr(\mathbf{V} = \mathbf{v} \,|\, \mathbf{E} = \mathbf{e})$.

The MPE-problem is the problem of finding a joint value assignment $\mathbf{v}$ to $\mathbf{V} \setminus \mathbf{E}$ such that $\Pr(\mathbf{v} \,|\, \mathbf{e})$ is maximal. The PARTIAL MAP-problem is the problem of finding a joint value assignment $\mathbf{v}$ to the so-called MAP-variables $\mathbf{V_{MAP}} \subsetneq \mathbf{V} \setminus \mathbf{E}$ such that $\Pr(\mathbf{v} \mid \mathbf{e})$ is maximal.

2.1 Complexity Theory

In the remainder, we assume that the reader is familiar with basic concepts of computational complexity theory, such as Turing Machines, the complexity classes P, NP, PP, #P, and completeness proofs for these classes. For a thorough introduction to these subjects we refer to textbooks like Garey and Johnson (1979) and Papadimitriou (1994). Furthermore, we use the concept of *oracle access*. A Turing Machine $\mathcal{M}$ has oracle access to languages in the class A, denoted as $\mathcal{M}^{\mathsf{A}}$, if it can query the oracle in one state transition, i.e., in $O(1)$. We can regard the oracle as a 'black box' that can answer membership queries in constant time. For example, $\mathsf{NP}^{\mathsf{PP}}$ is defined as the class of languages which are decidable in polynomial time on a non-deterministic Turing Machine with access to an oracle deciding problems in PP, like the well known INFERENCE-problem, which is PP-complete (Littman et al., 1998).

We will frequently use the fact that #P is polynomial-time Turing equivalent to PP (Simon, 1977). Informally, this implies that a class that uses #P as an oracle, can also be defined as using PP and vice versa. For example, the class $\mathsf{NP}^{\mathsf{PP}}$ is equal to the class $\mathsf{NP}^{\#\mathsf{P}}$; however, the former notation is more common. We will use this property frequently in our hardness proofs.

The complexity class P^{PP} is defined as the class of languages, decidable by a deterministic Turing Machine with access to a PP oracle. While P^{PP} is less known than the related

classes $\mathsf{NP}^{\mathsf{PP}}$ and $\mathsf{co}-\mathsf{NP}^{\mathsf{PP}}$, complete decision problems have been discussed in Toda (1994). Intuitively, while NP is associated with the *existence* of a satisfying solution, PP with a *threshold* of satisfying solutions, and #P with the *exact number* of satisfying solutions, P^{PP} is associated with the *middle* satisfying solution. For this class, the canonical complete problems Mid SAT and Kth SAT are the problems of determining whether in the lexicographically middle (k-th) satisfying assignment $x_1x_2\ldots x_n \in \{0,1\}^n$ to a Boolean formula ϕ, the least significant bit is odd (Toda, 1994).

The complexity results in this paper are based on *function*—rather than *decision*—problems. While a decision problem requires a *yes* or *no* answer (like 'Is there a satisfying truth assignment to the variables in a formula?'), a function problem requires a construct, like a satisfying truth assignment. Formally, traditional complexity classes like P and NP are defined on decision problems, using *acceptor* Turing Machines. The functional counterparts of these classes, like FP and FNP are defined using *transducer* Turing Machines; on an input x a transducer $\mathcal{M}$ *computes* y if $\mathcal{M}$ halts in an accepting state with y on its output tape. In our opinion, the problem of finding the k-th solution has a more 'natural' correspondence with function problems than decision problems and require less technical details in our hardness proofs.

To prove P^{PP} (or $\mathsf{FP}^{\mathsf{PP}}$) -hardness of a particular problem, one needs to reduce it from a known complete problem like Kth SAT. To prove *membership* of P^{PP} ($\mathsf{FP}^{\mathsf{PP}}$), one needs to show that it is accepted (computed) by a *metric Turing Machine*. Metric Turing Machines were defined by Krentel (1988).

Definition 1 (Metric Turing Machine). A metric Turing Machine (metric TM for short) is a polynomial-time bounded non-deterministic Turing Machine such that every computation path halts with a binary number on an output tape. Let $\hat{\mathcal{M}}$ denote a metric TM, then $\mathrm{Out}_{\hat{\mathcal{M}}}(x)$ denotes the set of outputs of $\hat{\mathcal{M}}$ on an input x, and $\mathrm{KthValue}_{\hat{\mathcal{M}}}(x,k)$ is defined to be the k-th smallest number in $\mathrm{Out}_{\hat{\mathcal{M}}}(x)$.

Toda showed (1994), that a function f is in $\mathsf{FP}^{\mathsf{PP}}$ if and only if there exists a metric TM $\hat{\mathcal{M}}$ such that f is polynomial-time one-Turing reducible[2] to $\mathrm{KthValue}_{\hat{\mathcal{M}}}$ ($f \leq^{\mathsf{FP}}_{1-T} \mathrm{KthValue}_{\hat{\mathcal{M}}}$ for short). Correspondingly, a set L is in P^{PP} if and only if a metric TM $\hat{\mathcal{M}}$ can be constructed, such that $\mathrm{KthValue}_{\hat{\mathcal{M}}}$ is odd for an input x if and only if $x \in L$. In the remainder, we will construct such metric TMs for the MPE- and Partial MAP-problems to prove membership in $\mathsf{FP}^{\mathsf{PP}}$ and $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$.

3 Enumerating MPE

In this section we will construct a $\mathsf{FP}^{\mathsf{PP}}$-completeness proof for the Kth MPE problem. More specifically, we show that Kth MPE can be computed by a metric TM in polynomial time (proving membership of $\mathsf{FP}^{\mathsf{PP}}$), and we prove hardness of the problem by a reduction from Kth SAT. We formally define the functional[3] version of Kth MPE problem as follows.

Kth MPE
Instance: Probabilistic network $\mathbf{B} = (\mathbf{G}, \Gamma)$, evidence variables $\mathbf{E}$ with instantiation $\mathbf{e}$, natural number k.
Question: What is the k-th most probable assignment $\mathbf{v_k}$ to the variables in $\mathbf{V} \setminus \mathbf{E}$ given evidence $\mathbf{e}$?

The functional version of Kth SAT, the problem that we will use in the reduction, is defined as follows.

Kth SAT
Instance: Boolean formula $\phi(x_1, \ldots, x_n)$, natural number k.
Question: What is the lexicographically k-th assignment $x_1 \ldots\ x_n \in \{0,1\}^n$ that satisfies ϕ?

We will use the formula $\phi_{ex} = ((x_1 \vee \neg x_2) \wedge x_3) \vee \neg x_4$ as a running example. We construct a

[2] A function f is *polynomial-time one-Turing reducible* to a function g if there exist polynomial-time computable functions T_1 and T_2 such that for every x, $f(x) = T_1(x, g(T_2(x)))$ (Toda, 1994, p.5).

[3] Note that we can transform this functional version into a decision variant by designating a variable $V_d \in \mathbf{V} \setminus \mathbf{E}$ with v_d as one of its values, and asking whether $V_d = v_d$ in $\mathbf{v_k}$.

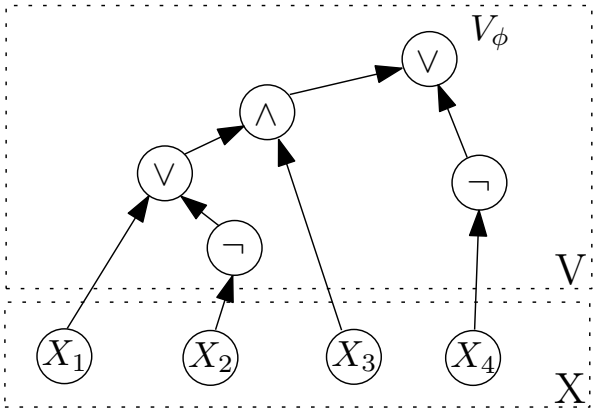


Figure 1: Example of k-th MPE construction for the formula $\phi_{ex} = ((x_1 \vee \neg x_2) \wedge x_3) \vee \neg x_4$

probabilistic network $\mathbf{B}_\phi$ from a given Boolean formula ϕ in the Kth SAT-instance with n variables x_i, as illustrated in Figure 1. For all variables x_i in the formula ϕ, we create a matching stochastic variable X_i in $\mathbf{V}$ for the network $\mathbf{B}_\phi$, with possible values *true* (T) and *false* (F). These variables are roots in the network $\mathbf{B}_\phi$ and are denoted as the *variable instantiation* part (X) of the network. The prior probabilities $p_1, \ldots, p_n$ for the variables $X_1, \ldots, X_n$ are chosen such that the prior probability of a particular value assignment $\mathbf{x}$ is higher than $\mathbf{x}'$, if and only if the corresponding truth assignment to $X_1, \ldots, X_n$ is lexicographically higher. More in particular, we choose prior probabilities $p_1, \ldots, p_i, \ldots, p_n$ such that $p_i = \frac{1}{2} - \frac{2^i - 1}{2^{n+1}}$. In our example with four variables, the probability distribution will be $p_1 = \frac{15}{32}$, $p_2 = \frac{13}{32}$, $p_3 = \frac{9}{32}$, and $p_4 = \frac{1}{32}$; the reader can verify that the probability of a value assignment $\mathbf{x}$ is higher than an assignment $\mathbf{x}'$, if and only if the corresponding truth assignment $x_1 \ldots\ x_n \in \{0,1\}^n$ is lexicographically smaller. Note that we can formulate these probabilities, using a number of bits which is polynomial in the input size.

For each logical operator in ϕ, we create an additional stochastic variable in the network, whose parents are the corresponding subformulas (or single sub-formula in case of a negation operator) and whose conditional probability table is equal to the truth table of that operator. For example, the variable corresponding to a $\wedge$-operator would have a conditional probability $\Pr(\wedge = T) = 1$ if and only if both its parents have the value *true*, and 0 otherwise. We denote the stochastical variable that is associated with the top-level operator in ϕ with V_ϕ. The thus constructed part of the network will be denoted as the *truth-setting* part (V) of the network. It is easy to see that, for a particular value assignment of variables X_i in the network, $\Pr(V_\phi = T) = 1$ if and only if the corresponding truth setting to the variables in ϕ satisfies ϕ.

Theorem 1. Kth MPE *is* $\mathsf{FP}^{\mathsf{PP}}$*-complete.*

Proof. To prove membership, we will show that a metric TM can be constructed for the Kth MPE-problem. Let $\hat{\mathcal{M}}$ be a metric nondeterministic TM that, on input $\mathbf{B}$, calculates $\Pr(\mathbf{V})$. Since $\Pr(\mathbf{V}) = \prod_{i=1}^{n} \Pr(v_i \mid \pi(V_i))$, $\hat{\mathcal{M}}$ calculates $\Pr(\mathbf{V} \mid \mathbf{e})$ by non-deterministically choosing instantiations v_i, consistent with evidence $\mathbf{e}$, at each step i, and multiplying the corresponding probabilities. The output is, for each computation path, a binary representation (e.g., in fixed precision notation) of $1 - \Pr(\mathbf{v} \mid \mathbf{e})$ with sufficient precision. Then, clearly $\text{KthValue}_{\hat{\mathcal{M}}}$ returns the k-th probable explanation of $\Pr(\mathbf{V} \mid \mathbf{e})$. This proves that Kth MPE is in $\mathsf{FP}^{\mathsf{PP}}$.

To prove hardness, we reduce Kth SAT to Kth MPE. Let ϕ be an instance of Kth SAT and let $\mathbf{B}_\phi$ be the network constructed from ϕ as described above. Observe that $\Pr(\mathbf{X} = \mathbf{x} \mid C = T) = 0$ if $\mathbf{x}$ represents a non-satisfying value assignment, and $\Pr(\mathbf{X} = \mathbf{x} \mid C = T)$ is equal to the prior probability of $\mathbf{X} = \mathbf{x}$ if $\mathbf{x}$ represents a satisfying value assignment. Furthermore note that the values of the variables that model logical operators are fully determined by the values of their parents. Then, given evidence $C = T$, the k-th MPE corresponds to the lexicographical k-th satisfying value assignment to the variables in ϕ. Thus, given an algorithm for calculating the k-th MPE, we can solve the Kth SAT problem as well. Clearly, the above reduction is a polynomial-time one-Turing reduction from Kth SAT to Kth MPE. This proves $\mathsf{FP}^{\mathsf{PP}}$-hardness of Kth MPE. □

Observe, that the problem remains $\mathsf{FP}^{\mathsf{PP}}$-complete when all nodes have indegree at most two, and all variables are binary.

4 Enumerating Partial MAP

While the MPE-problem is complete for the class NP (solvable by a nondeterministic TM), Partial MAP is complete for $\mathsf{NP}^{\mathsf{PP}}$, i.e., solvable by a nondeterministic TM with access to an oracle for problems in PP. In the previous section we have proven that the Kth MPE-problem is complete for $\mathsf{FP}^{\mathsf{PP}}$, thus solvable by a *metric* TM. Intuitively, this suggests that the Kth Partial MAP-problem is complete for $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$, the class of function problems solvable by a *metric* TM with access to a PP-complete oracle. To our best knowledge, no complete problems have been discussed for this complexity class. We introduce the Kth NumSat-problem, defined as follows.

Kth NumSAT
Instance: Boolean formula $\phi(x_1, \ldots, x_m, \ldots, x_n)$, natural numbers k, l.
Question: What is the lexicographically k-th assignment $x_1 \ldots\ x_m \in \{0,1\}^m$ such that exactly l assignments $x_{m+1} \ldots\ x_n \in \{0,1\}^{n-m}$ satisfy ϕ?

We will prove in the appendix that Kth NumSAT is $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete. To prove hardness of Kth Partial MAP, we will use a version of this problem with *bounds* on the probability of the MAP variables.

Kth Partial MAP
Instance: A probabilistic network $\mathbf{B} = (\mathbf{G}, \Gamma)$, evidence variables $\mathbf{E}$ with instantiation $\mathbf{e}$, observable variables $\mathbf{V_{MAP}} \subset \mathbf{V} \setminus \mathbf{E}$, natural number k, rational numbers $0 \leq q \leq r \leq 1$.
Question: What is, within the interval $[q, r]$, the k-th most probable assignment $\mathbf{v_k}$ to the variables in $\mathbf{V_{MAP}}$ given evidence $\mathbf{e}$?

Note that the Kth Partial MAP problem *without* boundary constraints is a special case where $q = 0$ and $r = 1$, and that we can use binary search techniques to find a solution to the *bounded* problem variant, using an algorithm for the *unbounded* problem variant, so we can transform a bounded problem variant into an unbounded problem variant in polynomial time, and vice versa. However, using the bounded problem formulation facilitates our hardness proof.

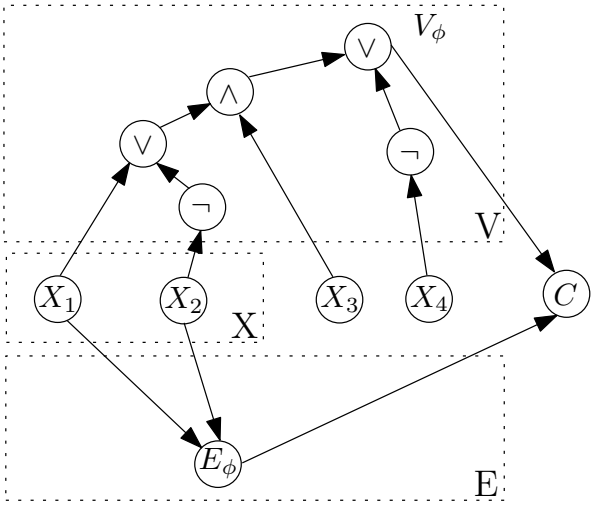


Figure 2: Example of k-th Partial MAP construction for the formula $\phi_{ex} = ((x_1 \vee \neg x_2) \wedge x_3) \vee \neg x_4$, with MAP variables x_1 and x_2

We will prove $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-completeness of Kth Partial MAP by a reduction from Kth NumSAT. We will again use the formula $\phi_{ex} = ((x_1 \vee \neg x_2) \wedge x_3) \vee \neg x_4$ as a running example (see Figure 2). We want to find the lexicographically k-th assignment to $\{x_1, x_2\}$ such that exactly l instantiations to $\{x_3, x_4\}$ satisfy ϕ_{ex}.

As in the previous section, we construct a probabilistic network $\mathbf{B}_\phi$ from a given Kth NumSAT instance $\phi(x_1, \ldots, x_m, \ldots, x_n)$. Again, we create a stochastical variable X_i for each variable x_i in ϕ, but now with uniform probability. We denote the variables $X_i, \ldots, X_m$ as the variable instantiation part (X). These variables are the MAP variables in our k-th Partial MAP construction. For each logical operator in ϕ, we create additional variables in the network as in the previous section, with V_ϕ as variable associated with the top level operator in ϕ. Observe that, for a particular value assignment $\mathbf{v_k}$ to the MAP variables

$\{X_1, \ldots, X_m\}$, $\Pr(V_\phi = T) = \frac{l}{n-m}$, where l is the number of value assignments to the variables $\{X_{m+1}, \ldots, X_n\}$ that satisfy ϕ.

Furthermore, we construct a *enumeration* part (E) of the network by constructing a $\log n$-deep binary tree with the MAP variables $X_1, \ldots, X_m$ as leafs and additional variables $E_{p,q}$, each with possible values *true* and *false*. Without loss of generality, we assume that the number of leafs is a power of two (we can use additional dummy variables). A variable $E_{p,1}$ has parents X_{2p-1} and X_{2p}; variables $E_{p,q}(q > 1)$ have parents $E_{2p-1,q-1}$ and $E_{2p,q-1}$. Let $\pi(E_{p,q}) = \{X_{2p-1}, X_{2p}\}$, respectively $\{E_{2p-1,q-1}, E_{2p,q-1}\}$ denote the parent configuration for $E_{p,1}$ and $E_{p,q}(q > 1)$. Then the conditional probability table for $E_{p,q}$ is defined as follows:

$$\begin{aligned}
\Pr(E_{p,q} = T \,|\, \pi(E_{p,q}) = \{T,T\}) &= 0 \\
\Pr(E_{p,q} = T \,|\, \pi(E_{p,q}) = \{T,F\}) &= \tfrac{1}{2^{p+n-m+1}} \\
\Pr(E_{p,q} = T \,|\, \pi(E_{p,q}) = \{F,T\}) &= \tfrac{2}{2^{p+n-m+1}} \\
\Pr(E_{p,q} = T \,|\, \pi(E_{p,q}) = \{F,F\}) &= \tfrac{3}{2^{p+n-m+1}}
\end{aligned}$$

The root of this tree will be denoted as E_ϕ. In the example network, there are only two MAP variables ($m = 2$) so $E_\phi = E_{1,1}$ with probabilities $\Pr(E_\phi = T) = 0, \frac{1}{16}, \frac{2}{16}$, and $\frac{3}{16}$ for the value assignments $\{T,T\}$, $\{T,F\}$, $\{F,T\}$ and $\{F,F\}$, respectively. Note that the above construct ensures that lexicographically smaller value assignments to the MAP variables, lead to a higher probability $\Pr(E_\phi = T)$, but that this probability is always less than $\frac{1}{2^{n-m}}$.

We add an additional variable C with parents V_ϕ and E_ϕ, with the following conditional probability table:

$$\Pr(C = T) = \begin{cases} 1 & \text{if } V_\phi = T \wedge E_\phi = T \\ \frac{1}{2} & \text{if } V_\phi = T \wedge E_\phi = F \\ \frac{1}{2} & \text{if } V_\phi = F \wedge E_\phi = T \\ 0 & \text{if } V_\phi = F \wedge E_\phi = F \end{cases}$$

We now have, that for a particular instantiation to the MAP variables, the probability $\Pr(C = T)$ is within the interval $[\frac{l}{2^{n-m}}, \frac{l+1}{2^{n-m}}]$, where l denotes the number of value assignments to the variables $X_{m+1}, \ldots, X_n$ that make ϕ true.

Theorem 2. KTH PARTIAL MAP *is* $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$*-complete.*

Proof. The $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$ membership proof is very similar to the $\mathsf{FP}^{\mathsf{PP}}$ membership proof of the KTH MPE-problem, but now we use an oracle for EXACT INFERENCE (which is #P-complete, see Roth (1996)) to compute the probability of the assignment $\mathbf{v_k}$. If it is within the interval $[q, r]$, we output 1 minus that probability; if not, we output 1. Note that we really need the oracle to perform this computation since we need to marginalize on $\mathbf{v_k}$. Clearly, $\text{KthValue}_{\hat{\mathcal{M}}}$ returns the k-th Partial MAP, and this proves that KTH PARTIAL MAP is in $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$.

To prove hardness, we construct a probabilistic network $\mathbf{B}_\phi$ from a given instance $\phi(x_1, \ldots, x_m, \ldots, x_n)$, similar to the previous section. The conditional probabilities in the thus constructed network ensure that the probability of a value assignment $\mathbf{v_k}$ to the variables $\{X_1, \ldots, X_m\}$ such that l value assignments to the variables $\{X_{m+1}, \ldots, X_n\}$ satisfy ϕ, is in the interval $[\frac{l}{2^{n-m}}, \frac{l+1}{2^{n-m}}]$. Moreover, $\Pr(C = T \mid \mathbf{x_k}) > \Pr(C = T \mid \mathbf{x'_k})$ if the truth value that corresponds with $\mathbf{x_k}$ is lexicographically smaller than $\mathbf{x'_k}$. Thus, with evidence $C = T$ and ranges $[\frac{l}{2^{n-m}}, \frac{l+1}{2^{n-m}}]$, the k-th Partial MAP corresponds to the lexicographical k-th truth assignment to the variables $x_1 \ldots x_m$ for which exactly l truth assignments to $x_{m+1} \ldots x_n$ satisfy ϕ. Clearly, the above reduction is a polynomial-time one-Turing reduction from KTH NUMSAT to KTH PARTIAL MAP. This proves $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-hardness of KTH PARTIAL MAP. □

Observe again, that the problem remains $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete when the MAP-variables have no incoming arcs, when all nodes have indegree at most two, and all variables are binary.

5 Conclusion

In this paper, we have addressed the computational complexity of finding the k-th MPE or k-th Partial MAP. We have shown that the KTH MPE-problem is P^{PP}-complete, making it considerably harder than both MPE (which

is NP-complete) and INFERENCE (which is PP-complete). The computational power (and thus the intractability of KTH MPE) of P^{PP} is illustrated by Toda's theorem (1991) that states that P^{PP} includes the entire Polynomial Hierarchy. Yet finding the k-th MPE is arguably easier than finding the most probable explanation given only partial evidence (the PARTIAL MAP-problem) which is $\mathsf{NP}^{\mathsf{PP}}$-complete. Moreover, when inference can be done in polynomial time (such as in polytrees) then we can find the k-th MPE in polynomial time (Sy, 1992; Srinivas and Nayak, 1996).

Finding the k-th Partial MAP, on the other hand, is considerably harder. We have shown that this problem is $\mathsf{P}^{\mathsf{PP}^{\mathsf{PP}}}$-complete in general. Park and Darwiche (2004) show that the PARTIAL MAP-problem remains NP-complete on polytrees, using a reduction from 3SAT[4]. Their proof can be easily modified to reduce KTH PARTIAL MAP on polytrees from the P^{PP}-complete problem KTH 3SAT (Toda, 1994), hence finding the k-th Partial MAP on polytrees remains P^{PP}-complete. Nevertheless, the approach of Park and Darwiche (2004) for approximating PARTIAL MAP may be extended to find the k-th Partial MAP as well.

For small or fixed k, these problems may be easier, depending on the exact problem formulation[5]. For example, it may be the case that KTH MPE is *fixed-parameter tractable*, i.e. an algorithm exists for KTH MPE which has a running time, exponentially only in k.

Acknowledgements

This research has been (partly) supported by the Netherlands Organisation for Scientific Research (NWO).
The author wishes to thank Hans Bodlaender and Gerard Tel for their insightful comments on earlier drafts of this paper, and Leen Torenvliet for discussions on the KTH NUMSAT problem.

[4]Technically, they reduce PARTIAL MAP from MAX SAT to preserve approximation results.

[5]The problem 'Are there *at least* k value assignments with a probability at least q' is trivially in NP for $k \leq \log n$, but when we want to know whether there are *exactly* k such assignments the problem may be considerable harder.

References

A. M. Abdelbar and S. M. Hedetniemi. 1998. Approximating maps for belief networks is NP-hard and other theorems. *Artificial Intelligence*, 102:21–38.

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 *Proceedings of the Second European Conference on AI and Medicine*, pages 247–256.

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 Fifteenth European Conference on Artificial Intelligence*, pages 675–679.

H. Chan and A. Darwiche. 2006. On the robustness of most probable explanations. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 63–71.

E. Charniak and S. E. Shimony. 1994. Cost-based abduction and map explanation. *Artificial Intelligence*, 66(2):345–374.

S. A. Cook. 1971. The complexity of theorem proving procedures. In *Annual ACM Symposium on Theory of Computing*, pages 151–158.

M. R. Garey and D. S. Johnson. 1979. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco.

F. V. Jensen. 2007. *Bayesian Networks and Decision Graphs.* Berlin: Springer Verlag, second edition.

M. W. Krentel. 1988. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36:490–509.

M. L. Littman, J. Goldsmith, and M. Mundhenk. 1998. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36.

C. H. Papadimitriou. 1994. *Computational Complexity.* Addison-Wesley.

J. D. Park and A. Darwiche. 2004. Complexity results and approximation settings for MAP explanations. *Journal of Artificial Intelligence Research*, 21:101–133.

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

D. Roth. 1996. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302.

E. Santos Jr. 1991. On the generation of alternative explanations with implications for belief revision. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 339–347.

S. E. Shimony. 1994. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410.

J. Simon. 1977. On the difference between one and many. In *Proceedings of the Fourth Colloquium on Automata, Languages, and Programming*, volume 52 of *LNCS*, pages 480–491. Springer-Verlag.

S. Srinivas and P. Nayak. 1996. Efficient enumeration of instantiations in Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pages 500–508.

B.K. Sy. 1992. Reasoning MPE to multiply connected belief networks using message-passing. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 570–576.

S. Toda. 1991. PP is as hard as the polynomial-time hierarchy. *SIAM Journal of Computing*, 20(5):865–877.

S. Toda. 1994. Simple characterizations of P(#P) and complete problems. *Journal of Computer and System Sciences*, 49:1–17.

J. Torán. 1991. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38(3):752–773.

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.

Appendix

In Section 4 we reduced Kth NumSAT to Kth Partial MAP. Here we show that Kth NumSAT is $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$-complete, and thus also $\mathsf{FP}^{\mathsf{PP}^{\mathsf{PP}}}$-complete.

Kth NumSAT
Instance: Boolean formula $\phi(x_1, \ldots, x_m, \ldots, x_n)$, natural numbers k, l.
Question: What is the lexicographically k-th assignment $x_1 \ldots x_m \in \{0,1\}^m$ such that exactly l assignments $x_{m+1} \ldots x_n \in \{0,1\}^{n-m}$ satisfy ϕ?

The hardness proof of Kth NumSAT is based on the $\mathsf{FP}^{\mathsf{PP}}$-hardness proof of Kth SAT by Toda (1994), and uses a result by Torán (1991) that states that the Counting Hierarchy (and thus $\mathsf{P}^{\mathsf{PP}^{\#\mathsf{P}}}$ in particular) is closed under polynomial time many-one reductions (and consequently, the functional counterpart $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$ is closed under polynomial time one-Turing reductions). Thus, any computation in $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$ can be modeled by a metric TM that calculates a bit string q based on its input x, then queries its #P oracle and writes down a number based on q and the result of the oracle, thus only querying the oracle once.

Theorem 3. Kth NumSAT *is* $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$*-complete.*

Proof. Since Toda's proof (Toda, 1994) relativizes, a function f is in $\mathsf{FP}^{\mathsf{PP}^{\#\mathsf{P}}}$ if there exists a metric TM $\hat{\mathcal{M}}$ with access to an oracle for #P-complete problems such that $f \leq^{\mathsf{FP}}_{1-T} \mathrm{KthValue}_{\hat{\mathcal{M}}}$. It is easy to see that a metric TM, that nondeterministically computes a satisfying assignment to $x_1 \ldots x_m$ (using an oracle for counting the number of satisfying assignments to $x_{m+1} \ldots x_n$), and writing the binary representation of this assignment on its output tape, suffices.

To prove hardness, let $\hat{\mathcal{M}}$ be a metric TM with a #P oracle. Given an input x to $\hat{\mathcal{M}}$, we can construct (using Cook's theorem (1971)) a tuple of two Boolean formulas $(\phi_x(q), \psi_x(r))$ such that ϕ_x is true if and only if q specifies a computation path of $\hat{\mathcal{M}}$ that is presented to the #P oracle, which returns the number l of satisfying instantiations to $\psi_x(r)$, such that $F(q,l)$ is the output of $\hat{\mathcal{M}}$. Since the computation path that computes q is uniquely determined, q is the k-th satisfying assignment to ϕ_x for which l instantiations to r satisfy $\psi_x(r)$, if and only if $F(q,l)$ is the k-th output of $\hat{\mathcal{M}}$. Thus, we can construct a $\leq^{\mathsf{FP}}_{1-T}$-reduction from every function accepted by a metric TM with access to a #P oracle to Kth NumSAT. □

Parameter Estimation in Mixtures of Truncated Exponentials

Helge Langseth
Department of Computer and Information Science
The Norwegian University of Science and Technology, Trondheim (Norway)
helgel@idi.ntnu.no

Thomas D. Nielsen
Department of Computer Science
Aalborg University, Aalborg (Denmark)
tdn@cs.aau.dk

Rafael Rumí and Antonio Salmerón
Department of Statistics and Applied Mathematics
University of Almería, Almería (Spain)
{rrumi,antonio.salmeron}@ual.es

Abstract

Bayesian networks with mixtures of truncated exponentials (MTEs) support efficient inference algorithms and provide a flexible way of modeling hybrid domains. On the other hand, estimating an MTE from data has turned out to be a difficult task, and most prevalent learning methods treat parameter estimation as a regression problem. The drawback of this approach is that by not directly attempting to find the parameters that maximize the likelihood, there is no principled way of e.g. performing subsequent model selection using those parameters. In this paper we describe an estimation method that directly aims at learning the maximum likelihood parameters of an MTE potential. Empirical results demonstrate that the proposed method yields significantly better likelihood results than regression-based methods.

1 Introduction

In domains involving both discrete and continuous variables, Bayesian networks with mixtures of truncated exponentials (MTE) (Moral et al., 2001) have received increasing interest over the last few years. Not only do MTE distributions allow discrete and continuous variables to be treated in a uniform fashion, but since the family of MTEs is closed under addition and multiplication, inference in an MTE network can be performed efficiently using the Shafer-Shenoy architecture (Shafer and Shenoy, 1990).

Despite its appealing approximation and inference properties, data-driven learning methods for MTE networks have received only little attention. In this context, focus has mainly been directed towards parameter estimation, where the most prevalent methods look for the MTE parameters minimizing the mean squared error w.r.t. a kernel density estimate of the data (Romero et al., 2006).

Although the least squares estimation procedure can yield a good MTE model in terms of generalization properties, there is no guarantee that the estimated parameters will be close to the maximum likelihood (ML) parameters. This has a significant impact when considering more general problems such as model selection and structural learning. Standard score functions for model selection include e.g. the Bayesian information criterion (BIC) (Schwarz, 1978), which is a form of penalized log-likelihood. However, the BIC score assumes ML parameter estimates, and since there is no justification for treating the least squares parameter estimates as ML parameters, there is in turn no theoretical foundation for using a least squared version of the BIC score.[1]

In this paper we propose a new parameter es-

[1]Learning the general form of an MTE can also be posed as a model selection problem, where we look for the number of exponential terms as well as appropriate split points. Hence, the problem also appears in this simpler setting.

timation method for univariate MTE potentials that directly aims at estimating the ML parameters for an MTE density with predefined structure (detailed below). The proposed method is empirically compared to the least squares estimation method described in (Romero et al., 2006), and it is shown that it offers a significant improvement in terms of likelihood.

The method described in this paper should be considered as a first step towards a general maximum likelihood-based approach for learning Bayesian networks with MTE potentials. Thus, we shall only hint at some of the difficulties (complexity-wise) that are involved in learning general conditional MTE potentials, and instead leave this topic as well as structural learning as subjects for future work.

2 Preliminaries

Throughout this paper, random variables will be denoted by capital letters, and their values by lowercase letters. In the multi-dimensional case, boldfaced characters will be used. The domain of the variable $\mathbf{X}$ is denoted by $\Omega_{\mathbf{X}}$. The MTE model 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 $\mathbf{W} = (W_1, \ldots, W_d)$ and $\mathbf{Z} = (Z_1, \ldots, Z_c)$ be the discrete and continuous parts of $\mathbf{X}$, respectively, with $c + d = n$. We say that a function $f : \Omega_{\mathbf{X}} \mapsto \mathbb{R}_0^+$ is a* Mixture of Truncated Exponentials (MTE) potential *if for each fixed value $\mathbf{w} \in \Omega_{\mathbf{W}}$ of the discrete variables $\mathbf{W}$, the potential over the continuous variables $\mathbf{Z}$ is defined as:*

$$f(\mathbf{z}) = a_0 + \sum_{i=1}^{m} a_i \exp\left\{\sum_{j=1}^{c} b_i^{(j)} z_j\right\}, \qquad (1)$$

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. We also say that f is an MTE potential if there is a partition $D_1, \ldots, D_k$ of $\Omega_{\mathbf{Z}}$ into hypercubes and in each D_i, f is defined as in Eq. (1).

An MTE potential is an *MTE density* if it integrates up to 1.

In the remainder of this paper we shall focus on estimating the parameters for a univariate MTE density. Not surprisingly, the proposed methods also immediately generalize to the special case of conditional MTEs having only discrete conditioning variables.

3 Estimating Univariate MTEs from Data

The problem of estimating a univariate MTE density from data can be divided into three tasks: *i*) partitioning the domain of the variable, *ii*) determining the number of exponential terms, and *iii*) estimating the parameters for a given partition of the domain and a fixed number of exponential terms. At this point we will concentrate on the estimation of the parameters, assuming that the split points are known, and that the number of exponential terms is fixed.

We start this section by introducing some notation: Consider a random variable X with density function $f(x)$ and assume that the support of $f(x)$ is divided into M intervals $\{\Omega_i\}_{i=1}^{M}$. Focus on one particular interval Ω_m. As a target density for $x \in \Omega_m$ we will consider an MTE with 2 exponential terms:

$$f(x|\boldsymbol{\theta}_m) = k_m + a_m e^{b_m x} + c_m e^{d_m x}, \; x \in \Omega_m. \quad (2)$$

This function has 5 free parameters, namely $\boldsymbol{\theta}_m = (k_m, a_m, b_m, c_m, d_m)$. For notational convenience we may sometimes drop the subscript m when clear from the context.

3.1 Parameter Estimation by Maximum Likelihood

Assume that we have a sample $\mathbf{x} = \{x_1, \ldots, x_n\}$ and that n_m of the n observations are in Ω_m. To ensure that the overall parameter-set is a maximum likelihood estimate for $\boldsymbol{\Theta} = \cup_m \boldsymbol{\theta}_m$, it is required that

$$\int_{x \in \Omega_m} f(x|\boldsymbol{\theta}_m)\, dx = n_m/n. \qquad (3)$$

Given this normalization, we can fit the parameters for each interval Ω_m separately, i.e., the parameters in $\boldsymbol{\theta}_m$ are optimized independently

of those in $\boldsymbol{\theta}_{m'}$. Based on this observation, we shall only describe the learning procedure for *a fixed interval* Ω_m, since the generalization to the whole support of $f(x)$ is immediate.

Assume now that the target density is as given in Eq. (2), in which case the likelihood function for a sample $\mathbf{x}$ is

$$L(\boldsymbol{\theta}_m|\mathbf{x}) = \prod_{i=1}^{n} \left\{ k_m + a_m e^{b_m x_i} + c_m e^{d_m x_i} \right\}. \quad (4)$$

To find a closed-form solution for the maximum likelihood parameters, we need to differentiate Eq. (4) wrt. the different parameters and set the results equal to zero. To exemplify, we perform this exercise for b_m, and obtain

$$\begin{aligned} \frac{\partial L(\boldsymbol{\theta}_m|\mathbf{x})}{\partial b_m} &= \sum_{i=1}^{n} \left\{ \frac{\partial L(\boldsymbol{\theta}_m|x_i)}{\partial b_m} \prod_{j \neq i} L(\boldsymbol{\theta}_m|x_j) \right\} \\ &= a_m b_m \sum_{i=1}^{n} e^{b_m x_i} \left\{ \prod_{j \neq i} \Big(k_m + a_m e^{b_m x_j} \right. \\ &\qquad \left. + c_m \, e^{d_m \, x_j} \Big) \right\}. \end{aligned} \quad (5)$$

Unfortunately, Eq. (5) is non-linear in the unknown parameters $\boldsymbol{\theta}_m$. Furthermore, both the number of terms in the sum as well as the number of terms inside the product operator grows as $O(n)$; thus, the maximization of the likelihood becomes increasingly difficult as the number of observations rise.

Alternatively, one might consider maximizing the logarithm of the likelihood, or more specifically a *lower bound* for the likelihood using Jensen's inequality. By assuming that $a_m > 0$ and $c_m > 0$ we have

$$\begin{aligned} \log\left(L(\boldsymbol{\theta}_m|\mathbf{x})\right) &= \sum_{i=1}^{n} \log\left(k_m + a_m \, \exp(b_m \, x_j) \right. \\ &\qquad \left. + c_m \, \exp(d_m \, x_j)\right) \\ &\geq \sum_{i=1}^{n} \log\left(k_m\right) + \sum_{i=1}^{n} \log\left(a_m \, \exp(b_m \, x_j)\right) \\ &\qquad + \sum_{i=1}^{n} \log\left(c_m \, \exp(d_m \, x_j)\right) \\ &= n\left[\log(k_m) + \log(a_m) + \log(c_m)\right] \\ &\qquad + (b_m + d_m) \sum_{i=1}^{n} x_i, \end{aligned} \quad (6)$$

and the idea would then be to maximize the lowerbound of Eq. (6) to push the likelihood upwards (following the same reasoning underlying the EM algorithm (Dempster et al., 1977) and variational methods (Jordan et al., 1999)). Unfortunately, though, restricting both a_m and c_m to be positive enforces too strict a limitation on the expressiveness of the distributions we learn.

Instead, an approximate solution can be obtained by solving the likelihood equations by numerical means. The proposed method for maximizing the likelihood is based on the observation that maximum likelihood optimization for MTEs can be seen as a constrained optimization problem, where constraints are introduced to ensure that both $f(x|\boldsymbol{\theta}_m) \geq 0$, for all $x \in \Omega_m$, and that Eq. (3) is fulfilled. A natural framework for solving this is the Lagrange multipliers, but since solving the Lagrange equations are inevitably at least as difficult as solving the unconstrained problem, this cannot be done analytically. In our implementation we have settled for a numerical solution based on Newton's method; this is described in detail in Section 3.1.2. However, it is well-known that Newton's method is quite sensitive to the initialization-values, meaning that if we initialize a search for a solution to the Lagrange equations from a parameter-set far from the optimal values, it will not necessarily converge to a useful solution. Thus, we need a simple and robust procedure for initializing Newton's method, and this is described next.

3.1.1 Naïve Maximum Likelihood in MTE Distributions

The general idea of the optimization is to iteratively update the parameter estimates until convergence. More precisely, this is done by iteratively tuning *pairs* of parameters, while the other parameters are kept fixed. We do this in a round-robin manner, making sure that all parameters are eventually tuned. Denote by $\hat{\boldsymbol{\theta}}_m^t = \left(k^t, a^t, b^t, c^t, d^t\right)$ the parameter values after iteration t of this iterative scheme. Algorithm 3.1 is a top-level description of this procedure, where steps 3 and 4 correspond to the optimization of the shape-parameters and steps

5 and 6 distribute the mass between the five terms in the MTE potential (the different steps are explained below).

Algorithm 3.1 ML estimation

1: Initialize $\hat{\theta}_m^0$; $t \leftarrow 0$.
2: **repeat**
3: $(a', b') \leftarrow \arg\max_{a,b} L(k^t, a, b, c^t, d^t \,|\, \mathbf{x})$
4: $(c', d') \leftarrow \arg\max_{c,d} L(k^t, a', b', c, d \,|\, \mathbf{x})$
5: $(k', a') \leftarrow \arg\max_{k,a} L(k, a, b', c', d' \,|\, \mathbf{x})$
6: $(k', c') \leftarrow \arg\max_{k,c} L(k, a', b', c, d' \,|\, \mathbf{x})$
7: $(k^{t+1}, a^{t+1}, b^{t+1}, c^{t+1}, d^{t+1},) \leftarrow (k', a', b', c, d')$
8: $t \leftarrow t + 1$
9: **until** convergence

For notational convenience we shall define the auxiliary function $p(s,t) = \int_{x \in \Omega_m} s \exp(tx)\, dx$; $p(s,t)$ is the integral of the exponential function over the interval Ω_m. Note, in particular, that $p(s,t) = s \cdot p(1,t)$, and that $p(1,0) = \int_{x \in \Omega_m} dx$ is the length of the interval Ω_m. The first step above is initialization. In our experiments we have chosen b^0 and d^0 as $+1$ and -1 respectively. The parameters k^0, a^0, and c^0 are set to ensure that each of the three terms in the integral of Eq. (3) contribute with equal probability mass, i.e.,

$$\begin{aligned} k^0 &\leftarrow \frac{n_m}{3n \cdot p(1,0)}, \\ a^0 &\leftarrow \frac{n_m}{3n \cdot p(1,b^0)}, \\ c^0 &\leftarrow \frac{n_m}{3n \cdot p(1,d^0)}. \end{aligned}$$

Iteratively improving the likelihood under the constraints is actually quite simple as long as the parameters are considered in pairs. Consider Step 3 above, where we optimize a and b under the constraint of Eq. (3) while keeping the other parameters (k^t, c^t, and d^t) fixed. Observe that if Eq. (3) is to be satisfied after this step we need to make sure that $p(a', b') = p(a^t, b^t)$. Equivalently, there is a functional constraint between the parameters that we enforce by setting $a' \leftarrow p(a^t, b^t)/p(1, b')$. Optimizing the value for the pair (a, b) is now simply done by line-search, where only the value for b is considered:

$$b' = \arg\max_b L(k, \frac{p(a^t, b^t)}{p(1,b)}, b, c^t, d^t | \mathbf{x}).$$

Note that at the same time we choose $a' \leftarrow p(a^t, b^t)/p(1, b')$. A similar procedure is used in Step 4 to find c' and d'.

Steps 5 and 6 utilize the same idea, but with a different normalization equation. We only consider Step 5 here, since the generalization is immediate. For this step we need to make sure that $\int_{x \in \Omega_m} k + a \exp(b'\, x)\, dx = \int_{x \in \Omega_m} k^t + a^t \exp(b'\, x)\, dx$, for any pair of parameter candidates (k, a). By rephrasing, we find that this is obtained if we insist that $k' \leftarrow k^t - p(a' - a^t, b')/p(1,0)$. Again, the constrained optimization of the pair of parameters can be performed using line-search in one dimension (and let the other parameter be adjusted to keep the total probability mass constant).

Note that Steps 3 and 4 do not move "probability mass" between the three terms in Eq. (2), these two steps only fit the shape of the two exponential functions. On the other hand, Steps 5 and 6 assume the shape of the exponentials fixed, and proceed by moving "probability mass" between the three terms in the sum of Eq. (2).

3.1.2 Refining the Initial Estimate

The parameter estimates returned by the line-search method can be further refined by using these estimates to initialize a nonlinear programming problem formulation of the original optimization problem. In this formulation, the function to be maximized is again the log-likelihood of the data, subject to the constraints that the MTE potential should be nonnegative, and that

$$g_0(\mathbf{x}, \boldsymbol{\theta}) \equiv \int_{x \in \Omega_m} f(x \,|\, \boldsymbol{\theta})\, dx - \frac{n_m}{n} = 0.$$

Ideally the nonnegative constraints should be specified for all $x \in \Omega_m$, but since this is not feasible we only encode that the function should be nonnegative in the endpoints e_1 and e_2 of the interval (we shall return to this issue later).

Thus, we arrive at the following formulation:

$$\begin{aligned}\text{Maximize } & \log L(\boldsymbol{\theta} \,|\, \mathbf{x}) = \sum_{i=1}^{n} \log L(\boldsymbol{\theta} \,|\, x_i) \\ \text{Subject to } & g_0(\mathbf{x}, \boldsymbol{\theta}) = 0, \\ & f(e_1 \,|\, \boldsymbol{\theta}) \geq 0, \\ & f(e_2 \,|\, \boldsymbol{\theta}) \geq 0,\end{aligned}$$

To convert the two inequalities into equalities we introduce slack variables:

$$f(x \,|\, \boldsymbol{\theta}) \geq 0 \Leftrightarrow f(x \,|\, \boldsymbol{\theta}) - s^2 = 0, \text{ for some } s \in \mathbb{R};$$

we shall refer to these new equalities using $g_1(e_1, \boldsymbol{\theta}, s_1)$ and $g_2(e_2, \boldsymbol{\theta}, s_2)$, respectively. We now have the following equality constrained optimization problem:

$$\begin{aligned}\text{Maximize } & \log L(\boldsymbol{\theta} \,|\, \mathbf{x}) = \sum_{i=1}^{n} \log L(\boldsymbol{\theta} \,|\, x_i) \\ \text{Subject to } & \mathbf{g}(\mathbf{x}, \boldsymbol{\theta}) = \begin{bmatrix} g_0(\mathbf{x}, \boldsymbol{\theta}) \\ g_1(e_1, \boldsymbol{\theta}, s_1) \\ g_2(e_2, \boldsymbol{\theta}, s_2) \end{bmatrix} = 0.\end{aligned}$$

This optimization problem can be solved using the method of Lagrange multipliers. That is, with the Lagrangian function $l(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{s}) = \log L(\boldsymbol{\theta} \,|\, \mathbf{x}) + \lambda_0 g_0(x, \boldsymbol{\theta}) + \lambda_1 g_1(x, \boldsymbol{\theta}, s_1) + \lambda_2 g_2(x, \boldsymbol{\theta}, s_2)$ we look for a solution to the equalities defined by

$$A(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{s}) = \nabla l(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{s}) = 0.$$

Such a solution can be found numerically by applying Newton's method. Specifically, by letting $\boldsymbol{\theta}' = (\boldsymbol{\theta}, s_1, s_2)^{\mathrm{T}}$, the Newton updating step is given by

$$\begin{bmatrix} \boldsymbol{\theta}'_{t+1} \\ \boldsymbol{\lambda}_{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}'_{t} \\ \boldsymbol{\lambda}_{t} \end{bmatrix} - \nabla A(\mathbf{x}, \boldsymbol{\theta}'_t, \boldsymbol{\lambda}_t)^{-1} A(\mathbf{x}, \boldsymbol{\theta}'_t, \boldsymbol{\lambda}_t),$$

where $\boldsymbol{\theta}'_t$ and $\boldsymbol{\lambda}_t$ are the current estimates and

$$A(\mathbf{x}, \boldsymbol{\theta}'_t, \boldsymbol{\lambda}_t) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}'} l(\mathbf{x}, \boldsymbol{\theta}', \boldsymbol{\lambda}) \\ \mathbf{g}(\mathbf{x}, \boldsymbol{\theta}') \end{bmatrix};$$

$$\nabla A(\mathbf{x}, \boldsymbol{\theta}'_t, \boldsymbol{\lambda}_t) = \begin{bmatrix} \nabla^2_{\boldsymbol{\theta}'\boldsymbol{\theta}'} l(\mathbf{x}, \boldsymbol{\theta}', \boldsymbol{\lambda}) & \nabla \mathbf{g}(\mathbf{x}, \boldsymbol{\theta}') \\ \nabla \mathbf{g}(\mathbf{x}, \boldsymbol{\theta}')^{\mathrm{T}} & 0 \end{bmatrix}.$$

As initialization values, $\boldsymbol{\theta}_0$, we use the maximum likelihood estimates returned by the line-search method described in Section 3.1, and in order to control the step size during updating, we employ the Armijo rule (Bertsekas, 1996). For the test results reported in Section 4, the Lagrange multipliers were initialized (somewhat arbitrarily) to 1 and the slack variables were set to $\sqrt{f(e_1 \,|\, \boldsymbol{\theta}_0)}$ and $\sqrt{f(e_2 \,|\, \boldsymbol{\theta}_0)}$, respectively.

Finally, it should be emphasized that the above search procedure may lead to $f(x \,|\, \boldsymbol{\theta})$ being negative for some x. In the current implementation we have addressed this problem rather crudely: simply terminate the search when negative values are encountered. Moreover, due to numerical instability, the search is also terminated if the determinant for the system is close to zero ($< 10^{-9}$) or if the condition number is large ($> 10^{9}$). Note that by terminating the search before convergence, we have no guarantees about the solution. In particular, the solution may be worse than the initial estimate. In order to overcome this problem, we always store the best parameter estimates found so far (including those found by line search) and return these estimates upon termination.

3.2 Parameter Estimation by Least Squares

Least squares (LS) estimation is based on finding the values of the parameters that minimize the mean squared error between the fitted model and the empirical density of the sample. In earlier work on MTE parameter estimation (Rumí et al., 2006), the empirical density was estimated using a histogram. In order to avoid the lack of smoothness, especially when data is scarce, (Romero et al., 2006) proposed to use kernels to approximate the empirical density instead of histograms.

As the LS method does not directly seek to maximize the likelihood of the model, the resulting LS parameters are not guaranteed to be close to the ML parameters. This difference was confirmed by our preliminary experiments, and has resulted in a few modifications to the LS method presented in (Rumí et al., 2006; Romero et al., 2006): *i*) Instead of us-

ing Gausian kernels, we used Epanechnikov kernels, which tended to provide better ML estimates in our preliminary experiments. *ii*) Since the smooth kernel density estimate assigns positive probability mass, p^*, outside the truncated region (called the boundary bias (Simonoff, 1996)), we reweigh the kernel density with $1/(1-p^*)$. *iii*) In order to reduce the effect of low probability areas, the summands in the mean squared error are weighted according to the empirical density at the corresponding points.

3.2.1 The Weighted LS Algorithm

In what follows we denote by $\mathbf{y} = \{y_1, \ldots, y_n\}$ the values of the empirical kernel for sample $\mathbf{x} = \{x_1, \ldots, x_n\}$, and with reference to the target density in Eq. (2), we assume initial estimates for a_0, b_0 and k_0 (we will later discuss how to get these initial estimates). With this outset, c and d can be estimated by minimizing the *weighted mean squared error* between the function $c\exp\{dx\}$ and the points $(\mathbf{x}, \mathbf{w})$, where $\mathbf{w} = \mathbf{y} - a_0\exp\{b_0\mathbf{x}\} - k_0$. Specifically, by taking logarithms, the problem reduces to linear regression:

$$\ln\{w\} = \ln\{c\exp\{dx\}\} = \ln\{c\} + dx,$$

which can be written as $w^* = c^* + dx$; here $c^* = \ln\{c\}$ and $w^* = \ln\{w\}$. Note that we here assume that $c > 0$. In fact the data $(\mathbf{x}, \mathbf{w})$ is transformed, if necessary, to fit this constraint, i.e., to be convex and positive. This is achieved by changing the sign of the values $\mathbf{w}$ and then adding a constant to make them positive. We then fit the parameters taking into account that afterwards the sign of c should be changed and the constant used to make the values positive should be subtracted.

A solution to the regression problem is then defined by

$$(c^*, d) = \arg\min_{c^*,d} \sum_{i=1}^{n} (w_i^* - c^* - dx_i)^2 y_i,$$

which can be described analytically:

$$c^* = \frac{(\sum_{i=1}^n w_i x_i y_i) - d\,(\sum_{i=1}^n x_i y_i)^2}{(\sum_{i=1}^n x_i y_i)}$$

$$d = \frac{\left(\sum_{i=1}^n w_i y_i\right)\left(\sum_{i=1}^n x_i y_i\right) - \left(\sum_{i=1}^n y_i\right)\left(\sum_{i=1}^n w_i x_i y_i\right)}{\left(\sum_{i=1}^n x_i y_i\right)^2 - \left(\sum_{i=1}^n y_i\right)\left(\sum_{i=1}^n x_i^2 y_i\right)}.$$

Once a, b, c and d are known, we can estimate k in $f^*(x) = k + ae^{bx} + ce^{dx}$, where $k \in \mathbb{R}$ should minimize the error

$$E(k) = \sum_{i=1}^{n} \frac{(y_i - ae^{bx_i} - ce^{dx_i} - k)^2 y_i}{n}.$$

This is achieved for

$$\hat{k} = \frac{\sum_{i=1}^n (y_i - ae^{bx_i} - ce^{dx_i}) y_i}{\sum_{i=1}^n y_i}.$$

Here we are assuming a fixed number of exponential terms. However, as the parameters are not optimized globally, there is no guarantee that the fitted model minimizes the *weighted mean squared error*. This fact can be somewhat corrected by determining the contribution of each term to the reduction of the error as described in (Rumí et al., 2006).

The initial values a_0, b_0 and k_0 can be arbitrary, but "good" values can speed up convergence. We consider two alternatives: *i*) Initialize the values by fitting a curve ae^{bx} to the modified sample by exponential regression, and compute k as before. *ii*) Force the empiric density and the initial model to have the same derivative. In the current implementation, we try both initializations and choose the one that minimizes the squared error.

4 Experimental Comparison

In order to compare the behaviour of both approaches we have used 6 samples of size 1000 taken from the following distributions: an MTE density defined by two regions, a beta distribution $Beta(0.5, 0.5)$, a standard normal distribution, a χ^2 distribution with 8 degrees of freedom, and a log-normal distribution $LN(0, 1)$.

	MTE	Beta	χ^2	Normal 2 splits	Normal 4 splits	Log-normal
ML	−2263.37	160.14	−2695.02	−1411.79	−1380.45	−1415.06
LS	−2307.21	68.26	−2739.24	−1508.62	−1403.46	−1469.21
Original LS	−2338.46	39.68	−2718.99	−1570.62	−1406.23	−1467.24

	MTE	Beta	χ^2	Normal 2 splits	Normal 4 splits	Log-normal
ML	−2263.13	160.69	−2685.76	−1420.34	−1392.28	−1398.3
LS	−2321.18	60.29	−2742.80	−1509.11	−1468.11	−2290.17
Original LS	−2556.68	39.42	−2766.86	−1565.28	−1438.67	−1636.99

Table 1: Comparison of ML vs. LS in terms of likelihood. In the upper table the split points were found using the method described in (Rumí et al., 2006), and in the lower table they were defined by the extreme points and the inflexion points of the exact density.

For the MTE, beta and normal distributions, we have used two split points, whereas for the log-normal and the χ^2 distributions, the number of splits points was set to 4. We have also run the experiment with four split points for the standard normal distribution.

The plots of the fitted models, together with the original density as well as the empirical histograms, are displayed in Figure 1. The split points used for these figures were selected using the procedure described in (Rumí et al., 2006).

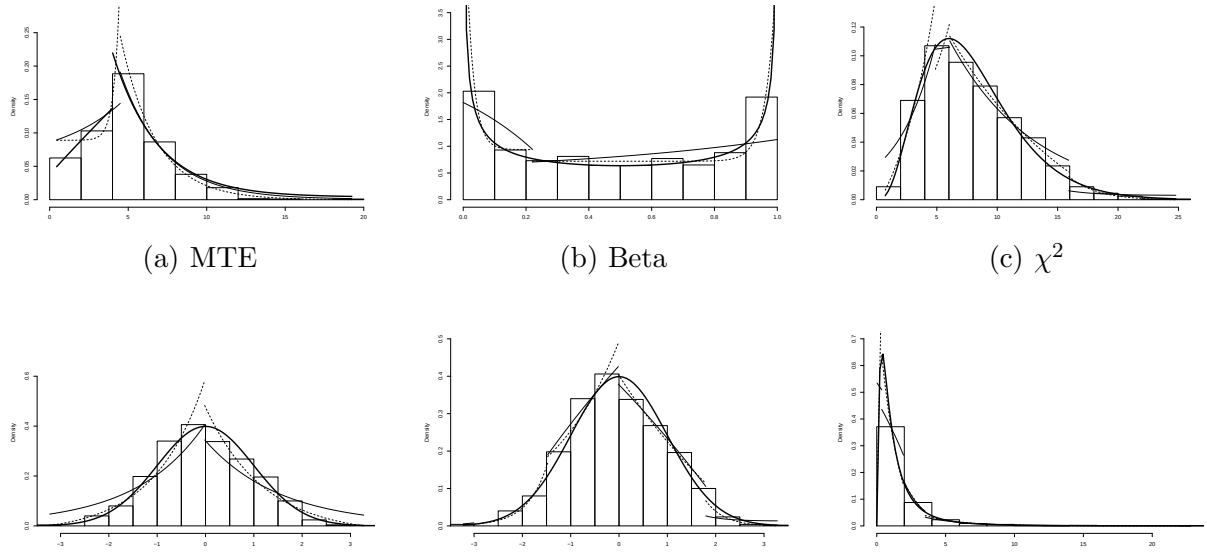
Table 1 shows the likelihood of the different samples for the models fitted using the direct ML approach, the modified LS method, and the original LS method described in (Rumí et al., 2006). The two sub-tables correspond to the split points found using the method described in (Rumí et al., 2006) and split points found by identifying the extreme points and the inflexion points of the of the true density, respectively.

From the results we clearly see that the ML-based method outperforms the LS method in terms of likelihood. This is hardly a surprise, as the ML method is actively using likelihood maximization as its target, whereas the LS methods do not. On the other hand, the LS and Original LS seem to be working at comparable levels. Most commonly (in 8 out of 12 runs), LS is an improvement over its original version, but the results for the Log-normal distribution (with the split-points selected according to the inflection points) cloud this picture; here the Original LS achieves a likelihood which is 10^{283} times as high as the one found by the LS method.

5 Conclusions and Future Work

In this paper we have introduced *maximum likelihood* learning for MTEs. Finding Maximum Likelihood parameters is interesting not only in its own right, but also as a vehicle to do more advanced learning: With maximum likelihood parameters we could, for instance, use the BIC criteria (Schwarz, 1978) to choose the number of exponential terms required to approximate the distribution function properly. We are currently in the process of evaluating this with the goal of avoiding overfitting during learning.

We are also considering to use a maximum likelihood-approach to learn *the location of the split-points*. Consider a sample $\mathbf{x} = \{x_1, \ldots, x_n\}$ where all samples are in the interval $\Omega_m = [\alpha, \beta)$, and assume the sample is sorted. A *brute force* approach to learning split-points could be to first fit MTE distributions on the intervals $[\alpha, (x_i + x_{i+1})/2)$ and $[(x_i + x_{i+1})/2, \beta)$, for each $i = 1, \ldots, n-1$, and calculate the likelihood of the data using the learned ML parameters. Then, one would choose the split-point, which gives the highest likelihood. Unfortunately, the complexity of this approach is squared in the sample size; we are currently investigating a number of simple heuristics to speed up the process. We have also started working on ML-based learning of *conditional distributions*, starting from the ideas published in (Moral et al., 2003). However, accurately locating the split-points for a conditional MTE is even more difficult than when learning marginals distributions; locating the split-



(a) MTE (b) Beta (c) χ^2

(d) Gaussian, 2 splits (e) Gaussian, 4 splits (f) Log-Normal

Figure 1: The plots show the results of samples from different distributions. The gold-standard distribution is drawn with a thick line, the MTE with Lagrange-parameters are given with the dashed line, and the results of the LS approach are given with the thin, solid line. The empiric distributions of each sample is shown using a histogram.

points for a variable X will not only influence the approximation of the distribution of X itself, but also the distributions for all the children of X.

Acknowledgments

This work has been supported by the Spanish Ministry of Education and Science, through project TIN2007-67418-C03-02.

References

D.P. Bertsekas. 1996. *Constrained optimization and Lagrange multiplier methods*. Academic Press Inc.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximun likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1 – 38.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Laurence K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233.

S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In *ECSQARU'01. Lecture Notes in Artificial Intelligence*, volume 2143, pages 135–143.

S. Moral, R. Rumí, and A. Salmerón. 2003. Approximating conditional MTE distributions by means of mixed trees. In *ECSQARU'03. Lecture Notes in Artificial Intelligence*, volume 2711, pages 173–183.

V. Romero, R. Rumí, and A. Salmerón. 2006. Learning hybrid Bayesian networks using mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 42:54–68.

R. Rumí, A. Salmerón, and S. Moral. 2006. Estimating mixtures of truncated exponentials in hybrid Bayesian network. *Test*, 15:397–421.

Gideon Schwarz. 1978. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464.

Glenn R. Shafer and Prakash P. Shenoy. 1990. Probability Propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–352.

J.S. Simonoff. 1996. *Smoothing methods in Statistics*. Springer.

An Anytime Algorithm for Evaluating Unconstrained Influence Diagrams

Manuel Luque
Dept. Inteligencia Artificial, UNED
28040 Madrid, Spain
mluque@dia.uned.es

Thomas D. Nielsen and Finn V. Jensen
Department of Computer Science
Aalborg University
9220 Aalborg, Denmark
{tdn,fvj}@cs.aau.dk

Abstract

Unconstrained influence diagrams (UIDs) extend the language of influence diagrams to cope with decision problems in which the order of the decisions is unspecified. Thus, when solving a UID we not only look for an optimal policy for each decision, but also for a so-called step-policy specifying the next decision given the observations made so far. However, due to the complexity of the problem temporal constraints can force the decision maker to act before the solution algorithm has finished, and, in particular, before an optimal policy for the first decision has been computed. This paper addresses this problem by proposing an anytime algorithm that computes a strategy and at any time provides a qualified recommendation for the first decisions of the problem. The algorithm performs a heuristic-based search in a decision tree representation of the problem. Experiments indicate that the proposed algorithm performs significantly better under time constraints than dynamic programming.

1 Introduction

An influence diagram (ID) is a framework for representing and solving Bayesian decision problems with a linear temporal ordering of decisions (Howard and Matheson, 1984). However, in many domains the process of finding an ordering of the decisions is an integral part of the decision problem, and in these situations the use of IDs would require all decision orderings to be explicitly specified in the model, possibly using artificial nodes and states. Examples of such decision problems include troubleshooting and medical diagnosis.

Unconstrained influence diagrams (UIDs) were introduced to represent and solve decision problems of this type (Jensen and Vomlelova, 2002); as a special case this also includes decision problems with a linear temporal ordering of the decisions. An optimal strategy in this framework consists not only of an optimal policy for each decision, but also of a step-strategy that prescribes the next decision to consider given the observations and decisions made so far. Such strategies are computable using dynamic programming in a way similar to that for traditional IDs (Shachter, 1986; Shenoy, 1992; Jensen et al., 1994; Madsen and Jensen, 1999).

Unfortunately, many real world problems have an inherent complexity that makes evaluation through exact methods intractable when time is scarce. Moreover, even if you had the time for solving the problem, storing the solution as a simple lookup table may be a problem: the number of possible past scenarios to consider in a policy can be intractably large. As

an example, the evaluation of the Ictneo system (Bielza et al., 1999) requires a table with $1,66 \times 10^{14}$ entries and produces a policy with $4,24 \times 10^{7}$ configurations for the first decision.

In this paper we present an anytime algorithm for solving UIDs. The algorithm provides a solution whenever it is stopped, and given sufficient time it will eventually provide a correct solution.

In comparison, the standard evaluation algorithm for UIDs (Jensen and Vomlelova, 2002) is a backward induction algorithm employing dynamic programming like most algorithms for IDs. It starts computing an optimal policy for the last decision and moves backwards in time until it reaches the first decision. If the process is stopped prematurely, the algorithm may provide a policy, however, the prescription for the first decision is completely un-informed. Furthermore, as described above, all effort so far may be spent on calculating a policy for a distant decision with an enormous space for the past; a task which will decrease considerably in size when you actually approach the point of the decision. If you consider a situation with a decision maker (DM) impatiently awaiting advice on what to do, he most probably wants to get an informed advice on the first decision rather than receiving detailed prescriptions for the last decisions.

To address this problem the proposed anytime algorithm starts with the first decision and works its way forward in time. Due to the nature of the problem, you cannot be sure of the policy for the first decision before the entire problem has been solved. However, the algorithm will over time gradually improve the probability of choosing the best decision.

2 Unconstrained Influence Diagrams

UIDs were proposed in (Jensen and Vomlelova, 2002) to represent decision problems in which the order of the decisions is not linear, and for which the DM is interested in the best ordering as well as an optimal choice for each decision.

2.1 The Representation Language

We start considering a very simple example: the diabetes diagnosis problem, introduced in (Demirer and Shenoy, 2001). A physician is trying to decide on a policy for treating patients. After an initial examination of their symptoms (S), the physician has to diagnose whether the patient is suffering from diabetes (D). Diabetes has two symptoms, glucose in urine and glucose in blood. Before deciding on whether or not to treat the patient for diabetes (Tr), the physician can decide to perform a urine test (UT) and/or a blood test (BT), which will produce the test results U and B, respectively. After the physician has observed the tests results (if any) she has to decide whether to treat the patient for diabetes. Observe that the order in which the tests are performed is not specified and that the result of a test is only available if the physician decides to perform the corresponding test.

To represent this problem by an influence diagram we have to represent the unspecified ordering of the tests as a linear ordering of decisions. This can be done by introducing two decision variables to model the first test and the second test, respectively. Unfortunately, the structure of the decision problem is not apparent from the model and for large decision problems this technique will be prohibitive as all possible scenarios should be explicitly encoded in the model.

In the UID framework, the combinatorial problem of representing non-sequential decision problems is postponed to the solution phase. A UID for the diabetes diagnoses problem is shown in Figure 1 (explained below).

More formally, an *unconstrained influence diagram* (UID) is a DAG over three sets of nodes: a set of decision nodes (rectangles) $\mathbf{V}_D$, chance nodes $\mathbf{V}_C$, and utility nodes (diamonds) $\mathbf{V}_U$. Chance nodes can be of two types, *observable* (circles) and *non-observable* (double-circles), and we require that utility nodes have no children. We will use the terms 'node' and 'variable' interchangeably if this does not cause any confusion.

The quantitative information associated with a UID consists of probability distributions and

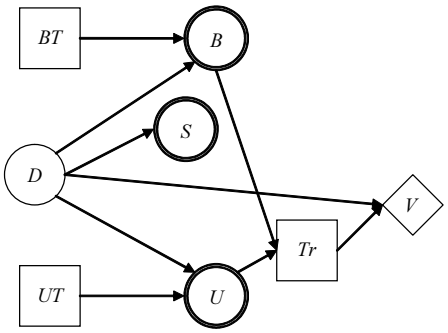


Figure 1: UID for the diabetes diagnosis problem.

utility functions. For each chance node C we have a probability distribution $P(C \mid \text{pa}(C))$ for C given its parents $\text{pa}(C)$, and for each utility node U we have a utility function ψ_U; ψ_U maps each configuration of the parents of U to a real number. We assume that the utility functions combine additively into a joint utility function ψ.

The semantics of the links are similar to the semantics from IDs, and the traditional no-forgetting assumption is also assumed. However, as opposed to IDs a total ordering of the decision nodes is not required. While non-observable variables are variables that will never be observed, an observable variable will be observed when all its antecedent decision variables have been decided. For example, in Figure 1 B is observed after deciding on BT, and S is observed before the first decision, since it has no antecedent decision variables.

The structural specification of a UID yields a partial temporal order. If a partial order is extended to a lineal order we get an influence diagram. Such an extended order is called an *admissible order*.

2.2 Solving a UID

Solving a UID means establishing a set of *step-policies* and a set of *decision-policies*. Together, the step-policies and the decision-policies form an optimal strategy. To organize the computations, we work with a secondary computational structure, called an S-DAG, which is a DAG representing the admissible orderings of the nodes in the UID (see Figure 2). A *GS-DAG* is a minimal S-DAG containing all admissible orderings relevant for computing an optimal strategy.

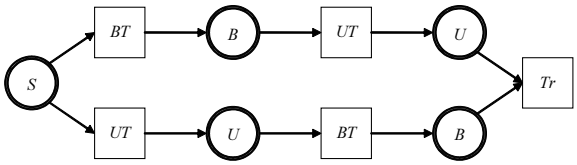


Figure 2: An S-DAG for the UID model of Figure 1.

A *step-policy* for a node N in an S-DAG is a rule that based on the current history $\text{hst}(N) \cup \{N\}$ specifies which of its children $\text{ch}(N)$ to go to. As the policy needs not be deterministic, we formally define a step-policy for node N as a conditional probability distribution $P(\text{ch}(N) \mid \text{hst}(N))$. A *decision-policy* for a decision node D in an S-DAG is probability distribution $P(D \mid \text{hst}(D))$. A *strategy* for an S-DAG consists of a step-policy for each node and a decision policy for each decision.

To define the expected utility (EU) of a strategy S, we unfold it to a *strategy tree*: following the policies of S we construct a tree in which all root-leaf paths represent admissible orderings. The expected utility for a strategy tree is defined as for decision trees, and it is by definition the expected utility of the strategy.

Jensen and Vomlelova (Jensen and Vomlelova, 2002) describe an algorithm for finding a strategy of maximum expected utility (MEU). The algorithm utilizes the S-DAG for the UID, and basically solves the UID/S-DAG through dynamic programming similarly to solving influence diagrams (i.e., eliminating the variables in reverse temporal order).

3 An Anytime Algorithm

In general, the basic idea with an anytime algorithm is that time constraints may cause the user to be unable to wait for the standard solution algorithm to finish. Thus, it should be possible to stop the algorithm at any time, and the

algorithm should then provide an approximate solution. With this requirement we may settle for an algorithm that may take longer than the standard algorithm, but which in the mean time can provide a better approximate solution than the standard algorithm.

With respect to UIDs, the standard algorithm provides a strategy by solving the problem in reverse temporal order. If the algorithm is stopped prematurely, it can provide a strategy, which consists of choosing completely randomly for the decisions which have not yet been dealt with, and to follow the calculated optimal policies for the last decisions. In this way, it can be said that you have an anytime algorithm; it provides a strategy whenever it is stopped, the expected utility of the strategy never decreases over time, and eventually, the algorithm provides an optimal strategy.

However, this is not satisfactory. If the user stops the algorithm prematurely, it is because she needs to take the first decision, but the algorithm does not give her any clue on what to do first. Therefore, the aim of an anytime algorithm for solving UIDs (or decision graphs in general) is to provide more and more informed advice on what to do first.

We propose an algorithm performing a forward search in a decision tree (Raiffa and Schlaifer, 1961) representation of the UID. The tree is built from the root toward the leaves, and it keeps a list of triggered nodes (the current leaves in the tree constructed so far) as candidates for expansion.[1] A triggered node X is *expanded* by adding its children to the tree and calculating the expected utility of the path from the root to X using a *heuristic function* for estimating the maximum expected utility (MEU) obtainable at the children of X.

3.1 A Search Based Solution Algorithm

An S-DAG (and a GS-DAG) can be converted into a decision tree (possibly using a dummy source node), which in turn can be used as a computational structure for solving the corresponding decision problem (disregarding complexity issues). A decision tree is a rooted tree in which the leaves are utility nodes and the nonleaf nodes are either decision nodes or chance nodes. The decisions on the possible orderings are made explicit in the model by partitioning the decision nodes into either ordinary decisions or branching point decisions.

The past of a node X (denoted by $\text{past}(X)$) is the configuration specified by the labels associated with the arcs on the path from the root to X; if X is a value node then $\text{past}(X)$ is called a *scenario*.

The quantitative part of the decision tree consists of probabilities and utilities. Each arc from a chance node A is associated with a probability $P(A = a \mid \text{past}(A))$, where $A = a$ is the label of the arc. These probabilities can be found by converting the UID into a Bayesian network: value nodes are removed, and decision nodes are replaced by chance nodes having no parents and with an arbitrary probability distribution.[2] Finally, with each value node V in the decision tree, we associate the utility $\psi(\text{past}(V))$ of the scenario $\text{past}(V)$. These utilities can be read directly from the UID model.

The decision tree represents each scenario in the decision problem explicitly; hence the size of the tree can grow exponentially in the number of variables. The size can, however, be reduced by collapsing identical subtrees, a procedure also know as *coalescence* (Olmsted, 1983). The opportunities for exploiting coalescence can be automatically detected in the S-DAG of the UID.

Instead of building the decision tree in full and solving it using the "average-out and fold-back" algorithm (Raiffa and Schlaifer, 1961), we propose to build the tree from the root toward the leaves. A heuristic function h provides an estimate of the MEU obtainable at every node in the decision tree. Thus, at any point in time we have a *partial decision tree* in which the heuristic can be used to estimate the MEU at the leaf nodes. These estimates can in turn

[1] The terminology is borrowed from AO* search algorithms (Nilsson, 1980), from which the proposed algorithm has been inspired.

[2] The time for computing the probabilities is small compared to the time required for evaluating the UID, and we shall therefore not consider this issue further.

be propagated upward in the tree, which gives an estimate of the MEU of the nodes in the explored part of the tree, and, in particular, an estimate of the optimal policy for the decision nodes in this part.

A collection of optimal policies for a subset of the decision nodes is called a *partial strategy* Δ', and a partial strategy based on the heuristic function is called a *partial heuristic strategy* $\hat{\Delta}'$. Clearly, the closer the heuristic function is at estimating the MEU of the triggered nodes in the partial decision tree, the closer the EU of $\hat{\Delta}'$ will be at the EU of Δ'.

A partial strategy can always be extended to a full (not necessarily optimal) strategy by assigning random policies to the decision nodes in the unexplored part of the tree. When we have a set of policies S, we define the *uniform extension* of S as a strategy Δ such that every policy in S is in Δ and the rest of the policies in Δ are uniform distributions.

3.2 Selecting a Heuristic Function

The choice of heuristic function not only determines the policies being computed, but it may in fact also be used to prune irrelevant parts of the tree thereby reducing complexity. A special class of heuristic functions are the so-called admissible heuristic functions.

Definition 1. A heuristic function h is said to be *admissible* if $h(N) \geq \text{MEU}(N)$ for any node N in the decision tree.

An admissible heuristic can be exploited during the search: Consider a decision node whose children X and Y are the roots in two subtrees. If the subtree defined by Y has been explored and $h(X) \leq \text{MEU}(Y)$, then we need not explore the subtree rooted at X.

Obviously, we would like the heuristic function h to define a tight upper bound on the expected utility, and relative to the computational complexity of solving the decision tree we would also like for h to be easy to compute.

3.2.1 An Admissible Heuristic

A possible choice of heuristic function could be (Vomlelova, 2003)

$$h_U(X) = \max_{l \in \mathcal{L}} \psi(\text{path}(X, l)), \qquad (1)$$

where $\mathcal{L}$ is the set of leaf nodes in the subtree rooted at X and $\psi(\text{path}(X, l))$ is the sum of the utilities associated with l and the path from X to l.

It is trivial to see that h_U is admissible. Moreover, h_U has the advantage of being computationally efficient, since it can be evaluated by max-marginalizing out the variables appearing in the domains of the utility potentials. The number of required max-marginalizations is at most $|\mathbf{V}_C \cup \mathbf{V}_D|$. In contrast to the dynamic programming approach, the complexity of computing this heuristic does not depend on the number of possible paths in the GS-DAG as max-operations commute.

Unfortunately, preliminary experiments have shown that h_U yields a very loose bound on the expected utility. For certain UIDs the estimated optimal policy for the first decision failed to stabilize over time, and in fact a random policy would on average provide a similar solution in terms of expected utility. Since we have not been able to define an alternative computationally efficient admissible heuristic, we have instead been looking for a nonadmissible heuristic.

3.2.2 A Nonadmissible Heuristic

The estimation given by the admissible heuristic h_U can be extremely far from the MEU. However, since it provides an upper bound on the expected utility, we can use it in combination with a lower bound to derive a good approximation to the expected utility.

As a lower bound h_L, we use the expected utility of the uniform extension of the current partial strategy; decision nodes in the unexplored part of the decision tree are treated as chance nodes with a uniform distribution. Relative to the computational complexity of solving the UID, this heuristic can be calculated efficiently by sum-marginalizing out the

variables in the utility and probability potentials. The number of required marginalizations is at most $|\mathbf{V}_C \cup \mathbf{V}_D|$ and does not depend on the number of paths in the GS-DAG as sum-marginalizations commute (this means that we are not required to follow an admissible elimination order consistent with the UID).

If all the variables in the future of node X are chance variables, i.e., if future$(X) \subseteq \mathbf{V}_C$, then $h_L(X) = \text{MEU}(X)$. Furthermore, as the number of decision nodes in future(X) increases the larger the difference $\text{MEU}(X) - h_L(X)$ will be. The opposite holds for the heuristic $h_U(X)$.

In order to derive a heuristic close to the actual expected utility, we define the nonadmissible heuristic h as a weighted linear combination of h_L and h_U:

$$h(X) = w_L(X)h_L(X) + w_U(X)h_U(X),$$

where $w_L(X) = \alpha \cdot k_X \cdot c(X)$ and $w_U(X) = \alpha \cdot d(X)$; here $c(X)$ and $d(X)$ are the number of chance and decision nodes in future(X), respectively, and α is a normalizing factor ensuring that $w_L(X) + w_U(X) = 1$. By varying the parameter k_X between 0 and $+\infty$, we can achieve any desired mixture of conservatism and optimism as defined by the two heuristics; note that k_X may be the same for all nodes.

One potential difficulty with this heuristic is how to choose a good value for k_X. To alleviate this problem, we propose to update k_X automatically as the tree is expanded. The intuition underlying the updating method is that we would in general expect the heuristic to be more precise the closer we get to the leaves: After a node X has been expanded we first estimate the expected utility of its children (using h and the current value for k_X). These estimates are then propagated upward in the tree: If X is a chance node, then the value propagated to X is $\widehat{\text{EU}}(X) = \sum_{Y \in \text{ch}(X)} P(Y|\text{past}(X))h(Y)$ and if X is a decision node then the value is $\widehat{\text{EU}}(X) = \max_{Y \in \text{ch}(X)} h(Y)$. By treating $\widehat{\text{EU}}(X)$ as an accurate estimate of the expected utility for X, we calculate a new value for k_X by setting $\widehat{\text{EU}}(X) = h(X)$:

$$k_X := \frac{\widehat{\text{EU}}(X) - \alpha h_U(X)d(X)}{\alpha c(X)h_L(X)}.$$

Note that k_X will always be non-negative, and that the update is not guaranteed to get us closer to the true expected utility (we might e.g. have started off with the "correct" value for k_X).

3.2.3 Performing the Search

The search/construction of the coalesced decision tree starts with the tree consisting of a single root node together with its children (such a tree stump is always uniquely identifiable). From this tree structure the method iteratively expands a node consistent with the UID specification.

When a node is expanded, its outgoing links are added to the decision tree as well as any successor node not already in the tree; the node to be expanded is always selected among the triggered nodes/leaves. When a node is added to the decision tree, a heuristic estimate of the MEU for that node is calculated. The values are then propagated upwards, possibly updating the current partial heuristic strategy.

The choice of which node to expand is nondeterministic. We have experimented with three selection schemes: (*i*) expand the node X with highest probability $P(\text{past}(X))$ of occurring (decision nodes are given an even probability distribution), (*ii*) expand the node X with highest weight $w(X) = P(\text{past}(X)) \cdot h(X)$, where h is the heuristic function estimating the expected utility of node X, and (*iii*) expand the node of lowest depth, i.e., perform a breadth first search. Preliminary experiments suggest that the latter provides the best results, and this is therefore the selection scheme used in the tests documented in Section 4.

4 Experiments

We have performed a series of experiments for assessing the performance of the proposed algorithm. For comparison we used dynamic programming (Jensen and Vomlelova, 2002), and to test the algorithms we generated a collection of random UIDs.

4.1 Generation of UIDs

It is easy to come up with artificial UID structures that, from a specification point of view, cannot be considered proper models of real-world decision problems. As an example, consider a UID with a decision node having only barren nodes (Shachter, 1986) in its future. Thus, rather than generating completely random UIDs (Vomlelova, 2003) we have instead tried to guide the UID generation by making perturbations of pre-specified UID templates.

Specifically, we manually constructed four UID templates from which we sampled 13 UID structures with varying number of decision nodes, chance nodes, and observable chance nodes. For each structure we randomly generated 50 realizations (probability and utility tables), producing a total of 650 models. Space restrictions prevent us from including additional details, but all models (including the templates) and a description of the sampling algorithm can be found at `www.ia.uned.es/~mluque/UID`.

4.2 Evaluation Metrics

The proposed anytime algorithm is intended for situations, where a DM is required to take one or more initial decisions but does not have time to wait for dynamic programming to finish. On the other hand, after the specified decisions have been taken we assume that there is sufficient time for dynamic programming to return an optimal strategy for the remaining decisions. Here we also assume that simply solving the UID offline and storing the policies as look-up tables is prohibitive due to space requirements.

The performance of the algorithm is evaluated according to the following two characteristics. *i*) The frequency with which the anytime algorithm returns the correct decision options (relative to the optimal strategy) for all decision nodes down to the ith level in the decision tree. *ii*) The expected utility of following the strategy prescribed by the anytime algorithm for the first i levels of decisions, followed by the optimal strategy for the remaining decisions. Both of these two measures depend on the amount of computation time used by the anytime algorithm, and to compare the results for different models, time is thus specified relative to the time required for dynamic programming to finish.

4.3 Experimental Results

The algorithms were implemented in Java 6.0 with the Elvira software package.[3] The experiments were performed on an Intel Core 2 computer (2.4 GHz) with 2 GB of memory.

First of all, it is important to emphasize that all reported values are normalized with the uniform strategy as baseline value, i.e., the uniform strategy and the optimal strategy attains the values 0 and 1, respectively.

The results obtained by letting the anytime algorithm run for e.g. 50% of the time required by dynamic programming are listed in the second column in Table 1; $EU^i(t)$ and $AccFreqDec^i(t)$ correspond to the two measures described in Section 4.2. In particular, $AccFreqDec^1(t)$ denotes the frequency of selecting the best initial decision (i.e., a branching point decision). For example, if we assume that the initial choice is between two decisions, then the anytime algorithm returns the optimal decision with a frequency of 0.742 $(0.5 + 0.484 \cdot 0.5)$. Similarly, suppose that the expected utility of following a random policy for the first decision is 90 and the MEU is 100, then a value of 0.514 for $EU^1(t)$ corresponds to an expected utility of 95.14.

From the results we clearly see that the algorithm improves over time w.r.t. all the recorded characteristics. Additional results can be found at `www.ia.uned.es/~mluque/UID`.

[3]The Elvira program was developed as a collaborative project of several Spanish universities (Elvira Consortium, 2002). The program and its source code can be downloaded from `www.ia.uned.es/~elvira`.

	25 %	**50 %**	**75 %**
$EU^1(t)$	0,442	0,514	0,538
$EU^2(t)$	0,609	0,769	0,865
$EU^3(t)$	0,546	0,703	0,794
$AccFreqDec^1(t)$	0,383	0,484	0,505
$AccFreqDec^2(t)$	0,396	0,503	0,563
$AccFreqDec^3(t)$	0,291	0,381	0,428

Table 1: Results for the anytime algorithm.

Acknowledgements

The first author was supported by the Department of Education of Madrid, the European Social Fund and the Spanish Ministry of Education and Science (grant TIN-2006-11152). We would like to thank Marta Vomlelova for giving us access to her UID implementation.

References

[Bielza et al.1999] C. Bielza, S. Ríos, and M. Gómez. 1999. Influence diagrams for neonatal jaundice management. In *AIMDM '99*, pages 138–142, London, UK. Springer-Verlag.

[Demirer and Shenoy2001] R. Demirer and P. P. Shenoy. 2001. Sequential valuation asymmetric decision problems. *Lecture Notes in Computer Science*, pages 252–265.

[Elvira Consortium2002] The Elvira Consortium. 2002. Elvira: An environment for creating and using probabilistic graphical models. In *PGM'02*, pages 1–11, Cuenca, Spain.

[Howard and Matheson1984] R. A. Howard and J. E. Matheson. 1984. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, pages 719–762.

[Jensen and Vomlelova2002] F. V. Jensen and M. Vomlelova. 2002. Unconstrained influence diagrams. In *UAI'02*, pages 234–241, San Francisco, CA. Morgan Kaufmann.

[Jensen et al.1994] F. Jensen, F. V. Jensen, and S. L. Dittmer. 1994. From influence diagrams to junction trees. In *UAI'94*, pages 367–373, San Francisco, CA. Morgan Kaufmann.

[Madsen and Jensen1999] A. Madsen and F. V. Jensen. 1999. Lazy evaluation of symmetric Bayesian decision problems. In *UAI'99*, pages 382–390, San Francisco, CA. Morgan Kaufmann.

[Nilsson1980] N. J. Nilsson. 1980. *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA.

[Olmsted1983] S. M. Olmsted. 1983. *On Representing and Solving Decision Problems*. Ph.D. thesis, Dept. Engineering-Economic Systems, Stanford University, CA.

[Raiffa and Schlaifer1961] H. Raiffa and R. Schlaifer. 1961. *Applied Statistical Decision Theory*. MIT press, Cambridge.

[Shachter1986] R. D. Shachter. 1986. Evaluating influence diagrams. *Operations Research*, 34:871–882.

[Shenoy1992] P. P. Shenoy. 1992. Valuation based systems for Bayesian decision analysis. *Operations Research*, 40:463–484.

[Vomlelova2003] M. Vomlelova. 2003. Unconstrained influence diagrams - experiments and heuristics. In *WUPES'2003*, Hejnice, Czech Republic.

An Independence of Causal Interactions Model for Opposing Influences

Paul P. Maaskant[1] & **Marek J. Druzdzel**[2,3]

[1] Department of Media and Knowledge Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands
[2] Faculty of Computer Science, Białystok Technical University, Wiejska 45A, 15-351 Białystok, Poland
[3] Decision Systems Laboratory, School of Information Sciences and Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA 15260, USA

Abstract

We introduce the DeMorgan gate, an Independence of Causal Interactions (ICI) model that is capable of modeling opposing influences, i.e., a mixture of positive and negative influences of parents on a child. The model is a noisy version of a conjunctive normal form of Boolean functions and is an extension and a combination of the popular Noisy-OR and Noisy-AND models, preserving their intuitive semantics. We report the results of a simple experiment testing the usefulness of the proposed model for elicitation of conditional probability distributions.

1 Introduction

Bayesian networks (BNs) (Pearl, 1988) offer a sound framework for reasoning in uncertain problem domains. A Conditional Probability Table (CPT) in a BN specifies the relation between a variable and its immediate predecessors (parents) in the graph. A fundamental problem of CPTs is their exponential growth in the number of parent variables. For nodes with more than a handful of parents, common in practical models, eliciting the CPT from human experts is daunting. So is learning it from data, as there are typically not enough cases to learn every distribution in a CPT reliably.

Independence of Causal Influences (ICI) models (see Díez and Druzdzel (2008) for a comprehensive review of the existing ICI models), provide a solution by assuming that parent variables cause the effect independently of each other. The benefit of this assumption is such that the number of required parameters is linear, rather than exponential, in the number of parent variables. One of the main practical limitations of the existing ICI models is that they cannot model opposing influences, i.e., combinations of influences that increase, and decrease the posterior probability of the child variable. Existing attempts to address this problem are, we believe, weak. In this paper, we propose a new ICI model based on a combination of OR and AND gates in a Conjunctive Normal Form (CNF) of a Boolean expression that combines opposing influences, yet retains a clear parametrization.

A few words about the notation. We will use uppercase letters to denote random variables (e.g., X) and lowercase letters to denote their states (e.g., x). Because all variables in this paper will be Boolean, a variable X will take only two states, x and $\overline{x}$. Bold uppercase letters will denote sets of random variables (e.g., $\mathbf{X}$) and bold lowercase letters (e.g., $\mathbf{x}$) will denote value assignments to sets of random variables. We will use $\Pr(X)$ to denote the probability distribution over a variable X.

2 Foundations

2.1 The Noisy-OR Model

The Noisy-OR model (Pearl, 1988; Henrion, 1989) is a probabilistic extension of the logical OR relation. Its variables, e.g., X, are binary and can be either *present*, denoted as x, or *absent*, denoted as $\overline{x}$. Each *present* parent event can independently produce the child effect. Noisy-OR's amechanistic property assumes that if none of the parent variables $X_1, ..., X_n$ are present, then neither is the child variable Y, i.e.,

$$\Pr(\overline{y}|\overline{x}_1, \ldots, \overline{x}_n) = 1 \; . \tag{1}$$

We define the probability that x_i produces y as

$$\Pr(y|\overline{x}_1, ..., x_i, \ldots, \overline{x}_n) = z_i \; . \tag{2}$$

The ICI assumption allows us to derive the probability of $\mathbf{X}$ producing y as

$$\Pr(y|\mathbf{X}) = 1 - \prod_{X_i = x_i} (1 - z_i) \; .$$

Repeating this process for all possible parent configurations gives us the CPT. Henrion (1989) extended the Noisy-OR model by introducing a leak probability z_L, yielding

$$\Pr(y|\mathbf{X}) = 1 - (1 - z_L) \prod_{X_i = x_i} (1 - z_i) \; . \tag{3}$$

The leak variable z_L represents the probability that the child variable is in its *present* state, even when all the parents are *absent*. An intuitive interpretation of leak is that it represents the effect of unmodeled causes of Y.

2.2 The Amechanistic Property

Amechanistic ICI models (Heckerman and Breese, 1996) are a subclass of ICI models that make two additional assumptions: (1) each variable has a *typical* state, referred to as the *distinguished* state. This is usually the *default* state for that variable. (2) If all parent variables are in their distinguished states, then so is the child variable. For example, if all possible causes of coughing are absent, then coughing is absent as well. In the course of elicitation, we can reduce the mental load needed to imagine a causal influence and estimate its strength by assuming that all other nodes are in the state that is not active and does not interfere with the cause that we are focusing on. The parameters for an amechanistic ICI model can be obtained by asking simple and clear questions and, effectively, such a model is particularly suited for parametrization by human experts.

In the Noisy-OR gate, assumption (2) is captured by Eq. 1, while Eq. 2 expresses the question needed for parameter elicitation. This enables us to directly elicit the probabilistic strength of the causal influence of a parent variable X_i on a child variable Y by asking simple questions, such as "What is the probability of coughing if a patient has pneumonia and no other factors that may cause coughing are present?" We believe that the unquestionable popularity of the Noisy-OR model is in part due to its amechanistic property. Some proposals for canonical gates do not have the amechanistic property and this, we believe, is their major weakness at the outset.

3 Causal Interactions

We describe below four fundamental types of causal interactions between an individual parent X and a child node Y.

Cause This is the most common type of interaction, modeled in the Noisy-OR gate: X is a causal factor and has a positive influence on Y. This influence, just as is the case in the Noisy-OR gate, does not need to be perfect. For example, smoking is quite likely a causal factor in lung cancer. Yet, incidence of lung cancer among smokers, while much larger than incidence of lung cancer among non-smokers, is still within a few percent. Hence, the conditional probability of lung cancer given that a person is a smoker is still fairly low.

The distinguished state of a cause is the state in which the cause has no effect on the child. For example, being a non-smoker has no effect on lung cancer.

Barrier This is a negated counterpart of a cause: X is a factor that decreases the probability of Y. For example, regular exercise decreases the probability of heart disease. While it is a well established factor with a negative influence on heart disease, it is unable by itself to prevent heart disease. One way of looking at a barrier is that it is dual to a cause: Absence of the barrier event is a causal factor for the child, i.e., $\overline{x}$ is a cause. One might go around the very existence of barriers in knowledge engineering by using negated versions of the variables that represent them. In the example above, one might define a variable *Lack of regular exercise*, which would behave as a cause of the variable *Heart disease*. This, however, might become cumbersome if *Regular exercise* participated in other interactions in a model. It might happen, for example, that it is a parent of both *Heart disease* and *Good physical shape*. Because *Regular exercise* decreases the probability of one and increases the probability of the other, barrier, which is a negated cause, is a useful modeling construct.

The distinguished state of a barrier is also the state in which the barrier has no effect on the child. For example, exercise may be thought as not influencing the risk of heart disease and it is the distinguished state in this interaction. We should point out here that the concept of a distinguished state is relative to an interaction and the same variable can have different distinguished states in different interactions that it participates in.

Requirement X is required for Y to be present. There are perfect requirements, such as being a female is a requirement for being pregnant but there are also requirements that are in practice not absolutely necessary. For example, a sexual intercourse is generally believed to be a requirement for pregnancy, but it is not a strict requirement, as pregnancy may be also caused by artificial insemination.

The distinguished state of a requirement is the state that is necessary for the effect to take place at all. For example, being a female is a requirement for becoming pregnant and it is the distinguished state in this interaction.

Inhibitor X inhibits Y. For example, rain may inhibit wild land fire or use of a condom during intercourse with an infected individual may inhibit contracting the HPV virus. Like in the other types of interactions, the parent may be imperfect in inhibiting the occurrence of the child. Fire may start even if there is rain and effectiveness of a condom in protecting from the HPV virus is only around 70%. Similarly to the relationship between causes and barriers, inhibitors are dual to requirements: Absence of an inhibitor event is a requirement for the child.

The distinguished state of an inhibitor is the state that has no effect on the child, i.e., the inhibiting factor being absent. For example, *Rain* is an inhibitor of *Wild land fire*. Its distinguished state is *No rain*, in which case the fire may happen.

4 The DeMorgan Model

We start with the deterministic version of the DeMorgan gate and later extend it to accommodate noise.

4.1 Deterministic DeMorgan Gate

Promoting Influences Promoting influences (causes and barriers) are modeled well by the Noisy-OR gate, which is a noisy version of the following Boolean formula

$$Y = X_1 \vee X_2 \vee \ldots \vee X_m \ , \tag{4}$$

where Xs stand for causes or barriers. The distinguished state of Y is *absent*.

Inhibiting Influences Presence of any inhibitor U_i is sufficient to cancel the child effect. We can express the effect that a set of inhibiting influences (requirements and inhibitors) have on Y by the following Boolean function

$$\overline{Y} \equiv U_1 \vee U_2 \vee \ldots \vee U_n \ , \tag{5}$$

where Us stand for requirements and inhibitors.

Eq. 5 is similar to Eq. 4 but now the parents cancel the child event instead of producing it. Variable Y is *absent* if at least a single U_i is present. We assume that the distinguished state

of Us is *absent*. However, contrary to our previous example, we assume the distinguished state of the child Y to be *present*. This assumption is based on common sense: We cannot cancel an event that is not present.

Combining Influences We can combine promoting and inhibiting influences by first applying one of De Morgan's laws to Eq. 5. We get

$$Y \equiv \overline{U}_1 \wedge \overline{U}_2 \wedge \ldots \wedge \overline{U}_n \; . \tag{6}$$

A logical proposition that combines (4) and (6) can only be true for a particular parent configuration if both (4) and (6) are also true for that same configuration. This implies that both equations form a conjunction

$$Y = (X_1 \vee X_2 \vee \ldots \vee X_m) \wedge \overline{U}_1 \wedge \overline{U}_2 \wedge \ldots \wedge \overline{U}_n \; .$$

We now have the logical proposition that we need in order to define the interaction for the combined model with Xs and Us and the child variable Y. Note that the proposition on the right hand-side of Eq. 4.1 is in the CNF. Because each of the conjuncts, save one, consists of a single variable, we can build a simple model to represent this proposition, using only a single OR and a single AND gate.

4.2 Modeling Uncertainty

Noise for Promoting Influences Noise for promoting influences is best modeled by mimicking the Noisy-OR gate, i.e., specifying a causal strength parameter v_i for each of the promoting influences and adding a leak parameter v_L. v_i is the probability of y given that parent i is not in its distinguished state and all other parents are in their distinguished states (please note that we need to choose a different distinguished state for causes and barriers). Eq. 3, the formula for leaky Noisy-OR gives the probability of y as a function of v_is and v_L (the formula uses zs instead of vs).

Noise for Inhibiting Influences The distinguished state of a parent that models an inhibiting influence is its *absent* state, i.e., when the parent is absent it is *certain not to inhibit the child event*, but when present, *it will inhibit the child event with some probability*. When a parent U_i is in its non-distinguished state, we assign a probability d_i that it will inhibit the child event. We include this uncertainty in the network by adding a noise variable W_i that has the following behavior

$$\Pr(w_i|U_i) = \begin{cases} 0 & \text{if } U_i = \overline{u}_i \\ d_i & \text{if } U_i = u_i \end{cases} \; ,$$

and the child variable Y is equivalent to the conjunction of variables $\overline{W}_1, \ldots, \overline{W}_n$, as given by Eq. 6. We have shown by De Morgan's laws that the Eq. 5 is the logical equivalent of Eq. 6. Eq. 3 gives us the probability of Y occurring

$$\Pr(y|\mathbf{U}) = \begin{cases} \prod_{u_i \in +\mathbf{u}} (1 - d_i) & \text{if } +\mathbf{u} \neq \emptyset \\ 1 & \text{if } +\mathbf{u} = \emptyset \end{cases} \; .$$

We define $+\mathbf{u}$ to be the subset of $\mathbf{U}$ that contains all parents that are in their non-distinguished states.

It makes little sense to ask for the effect of rain on a bonfire, when the latter is absent. By analogy, we cannot determine d_i directly if we assume that the distinguished state of Y is *absent*. Therefore, we determine d_i relative to an arbitrary set of promoting influences (with a joint effect v on Y) or even the leak parameter, although it seems that elicitation will be more reliable for larger values of v.[1] Suppose we know the effect of a promoting influence X_i, denoted as v_i, and the effect of both X_i and inhibiting influence U_j, denoted as q_j, i.e.,

$$\begin{aligned} p &= 1 - (1 - v_L)(1 - v_i) \; , \\ q_j &= (1 - (1 - v_L)(1 - v_i))(1 - d_j) \; . \end{aligned}$$

We have $d_j = 1 - q_j/p$.

Derivation of the CPT The total effect of simultaneous presence of noisy promoting and inhibiting causes in a leaky noisy DeMorgan gate can be combined into a CPT as follows:

$$\Pr(y|\mathbf{X}, \mathbf{U}) = (1 - (1 - v_L) \prod_{x_i \in +\mathbf{x}} (1 - v_i)) \prod_{u_j \in +\mathbf{u}} (1 - d_j) \; .$$

[1] Thus, there are combinatorially many questions that we can ask in order to obtain d_j, something not unheard of in probability elicitation.

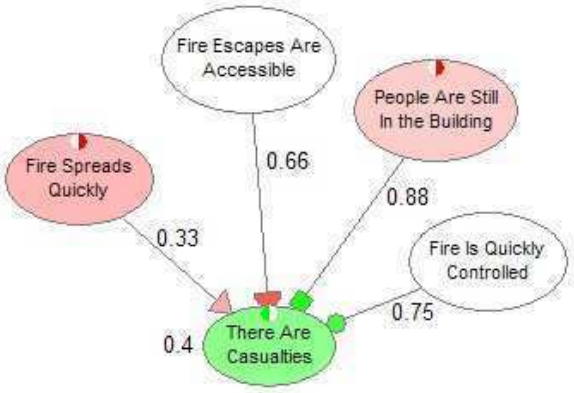


Figure 1: DeMorgan model example network

5 Knowledge Engineering for the DeMorgan Gate

A knowledge engineer has to most of all ensure that a gate elicited can be viewed as DeMorgan gate. The conditions that have to be fulfilled for the DeMorgan gate are similar to those listed for other canonical gates by Díez and Druzdzel (2008): Each parent must be able to cause or to inhibit the child node through a separate causal mechanism and there may be no significant interactions among these mechanisms.

Now, for each type of interaction, q_i, the parameter associated with the causal link from a parent X_i corresponds to the probability of the effect y happening if all parents but X_i are in their distinguished states. The leak parameter v_L expresses the probability of y given that all parents are in their distinguished states.

Consider the following network based on DeMorgan gate with one cause (*Fire Spreads Quickly*), one barrier (*Fire Escapes Are Accessible*), one requirement (*People Are Still In the Building*), and one inhibitor (*Fire Is Quickly Controlled*).

We will now give example questions to be asked of an expert. Please note that there is a natural discrepancy between what one has to say formally and what sounds clear to a human. Each of the questions listed below can be adjusted to the needs of particular context, i.e., their elements can be rephrased or omitted if they do not make sense.

The leak parameter "What is the probability of casualties if the fire does not spread quickly, fire escapes are not accessible, people are still in the building, and fire is not quickly controlled? Please note that casualties may happen due to other, unmodeled causes."

Cause "What is the probability of casualties if the fire spreads quickly, fire escapes are not accessible, people are still in the building, fire is not quickly controlled, and no other unmodeled causal factors are present?"

Barrier "What is the probability of casualties if the fire does not spread quickly, fire escapes are accessible, people are still in the building, fire is not quickly controlled, and no other unmodeled causal factors are present?"

Requirement "What is the probability of casualties if the fire does not spread quickly, fire escapes are not accessible, there are no people in the building, fire is not quickly controlled, and no other unmodeled causal factors are present?" Please note that the possible casualties are due to the fact that information concerning absence of people in the building may be false or the casualties may be that of the fire fighters.

Inhibitor "What is the probability of casualties if the fire does not spread quickly, fire escapes are not accessible, there are people in the building, fire is quickly controlled, and no other unmodeled causal factors are present?"

6 Empirical Evaluation

To validate the DeMorgan model, we conducted an experiment based on the methodology for evaluating probability elicitation schemes introduced by Wang et al. (2002). Its main advantage is that it controls for a-priori domain knowledge on the part of the subjects. The subjects are first asked to learn an abstract domain, which they have never seen before (typically an abstract interactive computer game). Since every subject may have a different set of experiences in the course of their interaction with the new domain, recording these provides us with a gold standard of the frequency

observed by the subject. A perfect elicitation scheme should retrieve these frequencies and the experimental setup aims at comparing elicitation schemes on how well they do so.

6.1 Subjects

Our subjects were 24 students in a graduate course *Decision Analysis and Decision Support Systems* in the School of Information Sciences, University of Pittsburgh. The students were familiar with, although not experts in, decision analysis, probability theory, and BNs. For their participation, they received a small course credit and a handful of M&Ms.

6.2 Experiment Design

The subjects were asked to play a simple, fictional computer game resembling a black box with four propositional inputs (X_1, X_2, X_3, and X_4) and one propositional output (Y) with states *Success* and *Failure*. Their task was to obtain *Success* at the output by means of selecting a combination of inputs. Subjects were allowed 160 trials, each trial consisting of three phases: (1) selecting values for X_1 through X_4, (2) pressing a key, and (3) observing the value of Y. The value of Y was chosen randomly by means of sampling from a DeMorgan gate, although the subjects were not aware of it. Two of the inputs (assigned randomly) were causes and the remaining two inputs were barriers. Model parameters were randomly chosen for each of the subject from the intervals $[0.5, 0.9]$ (causes), $[0.3, 0.9]$ (barriers) and $[0.1, 0.3]$ (the leak). Each subject faced thus a different probabilistic model driving the game.

Because of a relatively small number of subjects, we used a within-subject design. At the conclusion of the training phase, the subjects were asked (1) to give the full CPT ($\Pr(Y|X_1, X_2, X_3, X_4)$, consisting of 16 entries, and (2) indicate which inputs were promoting and which were inhibiting influences, assess their strengths and the leak probability. The order of the two elicitations was randomized to compensate for a possible carry-over effect.

6.3 Experiment Results

We used the probability distribution *observed* by each subject as the gold standard of what the subject knew. For each value OBS_i of the observed CPT, we calculated the maximum a-posteriori estimate given the subject's 160 observations, using a Beta prior distribution with a very small equivalent sample size (in order to avoid zero probabilities), i.e.,

$$OBS_i = \frac{s_i + 0.01}{t_i + 0.02} ,$$

where s_i denotes the number of successful trials, and t_i the total number of trials for input configuration i.

Of interest to us was the elicitation error, i.e., the difference between the observed CPT and the elicited probability distributions. We measured the error by the averaged Euclidean and Hellinger distances (Kokolakis and Nanopoulos, 2001). Because both measures are defined for single distributions, we averaged errors across all 16 distributions in the CPT.

The subjects each took between 30 and 45 minutes to complete the experiment. We judged one of the subjects to be an outlier, and excluded the subject from further analysis. This subject likely confused the concept of inhibiting with promoting, as she reported very low probabilities in cases where she observed very high probabilities and vice versa.

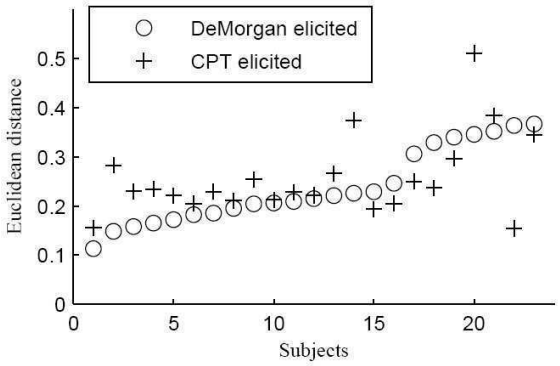


Figure 2: Raw data (sorted by increasing distance for the DeMorgan model)

Figure 2 shows raw data, i.e., the Euclidean distance for each subject: (1) the distance be-

tween the observed CPT and the CPT generated by the elicited DeMorgan model, sorted from the smallest to the largest distance, and (2) the distance between the observed CPT and the directly elicited CPT. We would like to point out that the range of distances is lower for the DeMorgan gate. Table 1 shows the aver-

Measure	DeMorgan	CPT
Averaged Euclidean Distance	0.2382	0.2566
Weighted Hellinger Distance	0.2481	0.2563

Table 1: Averaged Euclidean and Hellinger distances.

age Euclidean and Hellinger distances across all subjects. A one-tailed paired t-tests performed on both distance measures yielded $p \approx 0.14$ for the Euclidean, and $p \approx 0.29$ for the Hellinger distance, showing no significant difference in accuracy at the commonly used $\alpha = 0.05$ significance level. Although the accuracy gain in favor of the DeMorgan model was not statistically significant, our results suggest that the CPT generated by DeMorgan model is at least as accurate as a directly elicited CPT. This becomes a non-trivial advantage when the number of parent variables is larger. And so, for a family with 10 parent variables, we have 21 questions for the DeMorgan model, versus 1,024 questions needed to elicit the CPT directly.

7 Related Work

Inhibitors are mentioned by Pearl (1988), who calls them *global inhibitors* and lays the foundations for both requirements and inhibitors, as proposed in DeMorgan gate. Pearl stops short, however, from combining logical OR and AND gates with negation, which is what DeMorgan gate does.

Srinivas (1993) generalizes the Noisy-OR model to multiple states and proposed a model that is known as the "feeding lines model," embodying a world of possible functions that tie a node to its parents. It is quite likely that there exist functions among all possible that will combine positive and negative influences. Srinivas' proposal for an extension of Noisy-OR has never been adopted and we are not aware of any work extending the "feeding lines model."

Heckerman and Breese (1994) and later Lucas (2005) discuss the foundations of ICI models and draw attention to so called *decomposable* ICI models. Lucas analyzes in depth canonical models based on Boolean functions, reminding that there are 2^{2^n} different n-ary Boolean functions and so is the potential number of causal interactions. The DeMorgan model is decomposable, although it does not decompose into identical functions. It is indeed one of a huge number of possibilities, but as we argue in this paper, it may well be one of few that are intuitive and potentially readily adopted in practice ICI models.

A proposal for combining positive and negative influences has been the CAusal STrength (CAST) model (Chang et al., 1994), which is an extension of BNs that is able to model simultaneous opposing influences. Although very popular, particularly in government and military applications, a major weakness of the CAST model is its unclear parametrization. Parents can influence a child variable in both of their states and do not have a distinguished state, hence, are not amechanistic.

Lemmer and Gossink's *recursive Noisy-OR model* (Lemmer and Gossink, 2004) deals with positive and negative influences, although not in the same model, i.e., a model includes either all positive or all negative influences.

Finally, Xiang and Jia (Xiang and Jia, 2007) proposed a general model based on combining Noisy-AND gates with negation, apparently developed independently from this proposal. That model is capable of modeling positive and negative influences similarly to our proposal.

8 Conclusions

An important property of the DeMorgan model is that it is able to model any logical interaction between inputs, when their influences on the output are independent, i.e., when they are ICI. In particular, DeMorgan gate can handle a combination of positive and negative influences, while preserving both probabilistic soundness and the amechanistic property, critical in probability elicitation. Proba-

bilistic soundness ensures that it is mathematically correct, and propositional logic, that lies at its foundations, ensures that our model is meaningful and intuitive for humans.

The results of our experiment indicate that elicitation of a small DeMorgan model is at least as accurate as direct elicitation of a CPT. Yet, the DeMorgan model requires a number of parameters that is linear, rather than exponential, in the number of parent variables. We expect that the DeMorgan model will show a great advantage over direct elicitation especially for larger models. We have embedded the DeMorgan model in SMILE and QGeNIe, a qualitative interface to SMILE, our probabilistic reasoning engine, and made it available to the community (`http://genie.sis.pitt.edu/`). QGeNIe is useful in rapid modeling of problems involving propositional variables. We are currently working on extending the DeMorgan model to multi-valued variables along the lines of the Noisy-MAX and Noisy-MIN gates.

Acknowledgments

This work has been supported by the Air Force Office of Scientific Research grant FA9550-06-1-0243 and by Intel Research. All experimental data were obtained using SMILE, a Bayesian inference engine developed at the Decision Systems Laboratory and available at `http://genie.sis.pitt.edu/`. We would like to thank Adam Zagorecki for stimulating discussions on the topic of amechanistic ICI models and help with our experiment. We thank anonymous reviewers for the PGM–06 for valuable suggestion that improved the paper. Majority of this work was performed while the first author was at the Decision Systems Laboratory.

References

K.C. Chang, P.E. Lehner, A.H. Levis, Abbas K. Zaidi, and X. Zhao. 1994. On causal influence logic. Technical report, George Mason University, Center of Excellence for C3I.

F. Javier Díez and Marek J. Druzdzel. 2008. Canonical probabilistic models for knowledge engineering. Unpublished manuscript, available at http://www.ia.uned.es/∼fjdiez/papers/canonical.html.

David Heckerman and John S. 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 Mateo, CA. Morgan Kaufmann Publishers, Inc.

David Heckerman and John S. Breese. 1996. Causal independence for probability assessment and inference using Bayesian networks. *IEEE, Systems, Man, and Cybernetics*, 26:826–831.

Max 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.

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.

John 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 – Part B: Cybernetics*, 34(6):2252–2261.

Peter J.F. Lucas. 2005. Bayesian network modeling through qualitative patterns. *Artificial Intelligence*, 163(2):233–263.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Sampath Srinivas. 1993. A generalization of the noisy-OR model. In *Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI–93)*, pages 208–215, San Francisco, CA. Morgan Kaufmann Publishers.

Haiqin Wang, Denver H. Dash, and Marek J. Druzdzel. 2002. A method for evaluating elicitation schemes for probabilistic models. *Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(1):38–43.

Y. Xiang and N. Jia. 2007. Modeling causal reinforcement and undermining for efficient CPT elicitation. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1708–1718.

New Methods for Marginalization in Lazy Propagation

Anders L Madsen
HUGIN EXPERT A/S, Gasværksvej 5, DK-9000 Aalborg, Denmark
Anders.L.Madsen@hugin.com

Abstract

Even though existing algorithms for belief update in Bayesian networks (BNs) have exponential time and space complexity, belief update in many real-world BNs is feasible. However, in some cases the efficiency of belief update may be insufficient. In such cases minor improvements in efficiency may be important or even necessary to make a task tractable. This paper introduces two improvements to the message computation in Lazy Propagation (LP). We introduce one-step lookahead methods for sorting the operations involved in a variable elimination using Arc-Reversal (AR) and extend LP with the any-space property. The performance impacts of the methods are assessed empirically.

1 INTRODUCTION

There are two main reasons for the popularity of BNs (Pearl, 1988), (Cowell et al., 1999), and (Kjærulff and Madsen, 2008) as a formalism for modeling and reasoning with uncertainty: 1) a BN is an efficient and intuitive graphical representation of a joint probability distribution and 2) there exists tools implementing efficient algorithms for belief update.

As both exact and approximate belief update in general are *NP-hard* (Cooper, 1990b; Dagum and Luby, 1993), the use of exponential complexity algorithms is justified (unless P=NP). Even though existing algorithms for belief update have exponential time and space complexity, belief update on a large number of real-world BNs is feasible. However, in some cases the efficiency of belief update may be insufficient, but close to sufficient. In such cases minor improvements in efficiency may be important or even necessary to make a task tractable. Examples of such cases include analysis at the portfolio level in financial institutions where a belief update is performed for each customer. If the portfolio consists of 100000s of customers, then the time cost of belief update becomes an important issue and even a minor improvement in efficiency can have a large impact on the performance of the portfolio level analysis. In addition, the importance of belief update performance increases as the complexity of real-world BNs increases.

Most algorithms for exact belief update belongs to either the class of *query-based* or the class of *all-marginals* algorithms. The first class contains, for instance, Belief Propagation (Pearl, 1988), Arc-Reversal (AR) (Olmsted, 1983; Shachter, 1986), Symbolic Probabilistic Inference (SPI) (Shachter et al., 1990), Recursive Decomposition (RD) (Cooper, 1990a), Variable Elimination (VE) (Cannings et al., 1978; Zhang and Poole, 1996), Bucket Elimination (Dechter, 1996a), the Fusion operator (Shenoy, 1997), Query DAGs (Darwiche and Provan, 1997), Recursive Conditioning (RC) (Darwiche, 2000) and Value Elimination (VU) (Bacchus et al., 2003) while the latter class contains, for instance, Lauritzen-Spiegelhalter (Lauritzen and Spiegelhalter, 1988), HUGIN (Jensen et al., 1990), and Shenoy-Shafer (Shafer and Shenoy, 1990).

LP (Madsen and Jensen, 1999) combines query-based and all-marginals algorithms. Message passing is performed in a junction tree where clique and separator potentials are decomposed into sets of factors and messages are computed using a (revised) query-based algorithm in an attempt to exploit independence relations induced by evidence and barren vari-

ables. Recently, Madsen (2006) introduced LP algorithms where either AR, VE or SPI is used for message and marginal computation in a variable elimination based approach.

This paper introduces two improvements to message and marginal computation in LP: one-step look-ahead methods for sorting the AR operations involved in a variable elimination and a method to extend LP with the any-space property. We also report on the results of an empirical performance analysis.

2 PRELIMINARIES

A BN $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$ over variables $\mathcal{X}$ consists of an acyclic, directed graph (DAG) $\mathcal{G} = (\mathcal{X}, \mathsf{E})$ and a set of conditional probability distributions (CPDs) $\mathcal{P}$. It induces a joint probability distribution over $\mathcal{X}$ s.t. $\mathsf{P}(\mathcal{X}) = \prod_{\mathsf{X}\in\mathcal{X}} \mathsf{P}(\mathsf{X}|\mathrm{pa}(\mathsf{X}))$.

We consider *belief update* as the task of changing beliefs due to changes in the world manifested through observations. It is the task of computing the posterior marginal $\mathsf{P}(\mathsf{X} \mid \epsilon)$ for each $\mathsf{X} \in \mathcal{X}$. Evidence $\epsilon = \{\epsilon_1, \ldots, \epsilon_n\}$ consists of a set of variable instantiations. We let ϵ_X denote the instantiation of X, i.e., $\epsilon_\mathsf{X} = \{\mathsf{X} = \mathsf{x}\}$ and $\epsilon_\mathsf{X} \in \epsilon$, and let $\mathcal{X}_\epsilon$ denote the set of variables instantiated by evidence ϵ.

Definition 2.1 [Barren Variable]
A variable X is a *barren* w.r.t. a set $\mathsf{T} \subseteq \mathcal{X}$, evidence ϵ, and DAG $\mathcal{G}$, if $\mathsf{X} \notin \mathsf{T}$, $\mathsf{X} \notin \mathcal{X}_\epsilon$ and X only has barren descendants in $\mathcal{G}$ (if any).

A probability *potential* (Shafer and Shenoy, 1990) is a non-negative and not-all-zero function over a set of variables while a probability distribution is a potential that sums to one. For probability potential ϕ with *domain* $\mathrm{dom}(\phi) = \{\mathsf{X}_1, \ldots, \mathsf{X}_n\}$, we let $\mathrm{H}(\phi)$ denote the *head* (i.e., the conditioned variables) and $\mathrm{T}(\phi)$ denote the *tail* (i.e., the conditioning variables) of ϕ.

The *domain graph* $\mathcal{G}(\Phi) = (\mathcal{X}, \mathsf{E})$ of a set of probability potentials Φ over variables $\mathcal{X}$ is the graph spanned by $\mathcal{X}$ where for each ϕ an undirected edge is added between each pair of variables $\mathsf{X}, \mathsf{Y} \in \mathrm{H}(\phi)$ and a directed edge is added from each $\mathsf{X} \in \mathrm{T}(\phi)$ to each $\mathsf{Y} \in \mathrm{H}(\phi)$. We let $\mathrm{dom}(\Phi)$ denote the set of domain variables of potentials in Φ. The notion of barren variables can be extended to graphs with both directed and undirected edges (Madsen, 2006).

Definition 2.2 [Query]
A *query* on a set of probability potentials Φ is a triple $\mathsf{Q} = (\mathsf{T}, \Phi, \epsilon)$ where $\mathsf{T} \subseteq \mathcal{X}$ is the target.

The set $\mathsf{E} = \mathrm{dom}(\Phi) \setminus \mathsf{T}$ is referred to as the *elimination set*. The set of potentials Φ^* obtained by eliminating $\mathrm{dom}(\Phi) \setminus \mathsf{T}$ from Φ s.t. $\prod_{\phi\in\Phi^*} \phi = \mathsf{P}(\mathsf{T}, \epsilon_1 \mid \epsilon_2)$ where $\epsilon = \epsilon_1 \cup \epsilon_2$ is a *solution* to query Q. Notice that a query may have multiple solutions as a solution is a decomposition of the joint potential over target T. We define $\Phi_\mathsf{X} \subseteq \Phi$ as $\Phi_\mathsf{X} = \{\phi \in \Phi : \mathsf{X} \in \mathrm{dom}(\phi)\}$.

2.1 SOLVING QUERIES

Query-based belief update algorithms solve a single query $\mathsf{Q} = (\mathsf{T}, \Phi, \epsilon)$. In the following we assume that Y is to be eliminated in the process of solving Q. AR performs a sequence ρ of arc-reversal operations to make Y barren prior to removing its potential from Φ. Let X be a variable with parent set $\mathrm{pa}(\mathsf{X}) = \{\mathsf{Y}, \mathsf{X}_1, \ldots, \mathsf{X}_n\}$ and let $\mathrm{pa}(\mathsf{Y}) = \{\mathsf{X}_1, \ldots, \mathsf{X}_n\}$. An AR operation on arc (Y, X) is performed as follows:

$$\begin{aligned}&\mathsf{P}(\mathsf{X}|\mathsf{X}_1, \ldots, \mathsf{X}_n)\\ &= \sum_\mathsf{Y} \mathsf{P}(\mathsf{X}|\mathsf{Y}, \mathsf{X}_1, \ldots, \mathsf{X}_n)\mathsf{P}(\mathsf{Y}|\mathsf{X}_1, \ldots, \mathsf{X}_n), \quad (1)\\ &\mathsf{P}(\mathsf{Y}|\mathsf{X}, \mathsf{X}_1, \ldots, \mathsf{X}_n)\\ &= \frac{\mathsf{P}(\mathsf{X}|\mathsf{Y}, \mathsf{X}_1, \ldots, \mathsf{X}_n)\mathsf{P}(\mathsf{Y}|\mathsf{X}_1, \ldots, \mathsf{X}_n)}{\mathsf{P}(\mathsf{X}|\mathsf{X}_1, \ldots, \mathsf{X}_n)}. \quad (2)\end{aligned}$$

The AR-operation corresponds to arc-reversal in $\mathcal{G}(\Phi)$. It is necessary to avoid making cycles in the process of reversing arcs to eliminate Y. If $\mathrm{pa}(\mathsf{X}) \setminus \{\mathsf{Y}\} \neq \mathrm{pa}(\mathsf{Y})$, then we perform straightforward domain extensions.

Using VE Y is eliminated from Φ by marginalization of Y over the combination of potentials of Φ_Y and setting $\Phi^* = \Phi \setminus \Phi_\mathsf{Y} \cup \{\phi_\mathsf{Y}\}$ where $\phi_\mathsf{Y} = \sum_\mathsf{Y} \prod_{\phi\in\Phi_\mathsf{Y}} \phi$.

There is a rich literature on any-space algorithms, e.g., (Dechter, 1996b; Darwiche, 2000; Bacchus et al., 2003). We consider VU and RC. RC is an any-space algorithm for exact query-based belief update based on recursive conditioning (Darwiche, 2000). RC is an instantiation of the family of algorithms referred to

as VU (Bacchus et al., 2003). A main difference is that VU supports a dynamic conditioning order whereas the order is fixed in RC. RD (Cooper, 1990a), on the other hand, is a divide-and-conquer method that recursively decomposes the network and maps the resulting decomposition into a corresponding equation.

2.2 ALL-MARGINALS

All-marginals-based belief update algorithms solve a single query $Q = (\{X\}, \Phi, \epsilon)$ for each $X \in \mathcal{X}$. The *all-marginals* problem is usually solved by local procedures operating on a secondary computational structure known as the *junction tree* (also known as a join tree and a Markov tree) representation of the BN (Jensen and Jensen, 1994). Let $\mathcal{T}$ denote a junction tree with cliques $\mathcal{C}$ and separators $\mathcal{S}$. The cliques $\mathcal{C}$ are the nodes of $\mathcal{T}$ whereas the separators $\mathcal{S}$ annotate the links of $\mathcal{T}$. Each clique $C \in \mathcal{C}$ represents a maximal complete subgraph in an undirected graph[1] $\mathcal{G}^{\mathcal{T}}$. The link between two neighboring cliques A and B is annotated with the intersection $S = A \cap B$, where $S \in \mathcal{S}$.

Once $\mathcal{T}$ is constructed the CPD of each $X \in \mathcal{X}$ is associated with a clique C s.t. $\mathrm{fa}(X) \subseteq C$ where $\mathrm{fa}(X) = \{X\} \cup \mathrm{pa}(X)$. We let Φ_C denote the set of CPDs associated with $C \in \mathcal{C}$. Belief update proceeds as a two phase process where information is passed as messages between cliques over separators in two steps. Two messages are passed over each $S \in \mathcal{S}$; one message in each direction. Once the message passing process has completed, the marginal of each $X \in \mathcal{X}$ is computed from any node in $\mathcal{T}$ including X.

Algorithms such as HUGIN, Shenoy-Shafer, Lauritzen-Spiegelhalter, and LP differ w.r.t. the representation of clique and separator potentials and the computation of messages.

3 LAZY PROPAGATION

Message passing proceeds according to the Shenoy-Shafer scheme: A clique A sends a message $\Phi_{A \to B}$ to its neighbor B when it has received messages from all neighbors (denoted $\mathrm{ne}(A)$) except B, see Figure 1. A message $\Phi_{A \to B}$ is the solution to a query $Q = (B, \Phi_A \cup \bigcup_{C \in \mathrm{ne}(A) \setminus B} \Phi_{C \to A}, \epsilon)$ and it is computed as:

$$\Phi_{A \to B} = \Big(\Phi_A \cup \bigcup_{C \in \mathrm{ne}(A) \setminus \{B\}} \Phi_{C \to A}\Big)^{M \downarrow B},$$

where M is the marginalization algorithm, i.e., either AR, VE or SPI. Prior to applying M to solve Q, potentials for which all head variables are barren and potentials over variables which are all separated from B given ϵ in $\mathcal{G}(\Phi_A \cup \bigcup_{C \in \mathrm{ne}(A) \setminus B} \Phi_{C \to A})$ are removed. Notice that $\Phi_{A \to B}$ and Φ_C are sets of potentials. The content of $\Phi_{A \to B}$ depends on M.

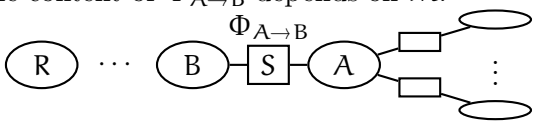


Figure 1: $\Phi_{A \to B}$ is passed from A to B.

The decomposition of potentials and the lazy elimination of variables enable an efficient exploitation of independence relations and barren variables during belief update. LP uses the structure of $\mathcal{T}$ to define a partial order on the elimination of variables in the computation of $P(X \mid \epsilon)$ for each $X \in \mathcal{X}$. While the domain of $\Phi_{A \to B}$ is defined by the elimination set $E = A \setminus B$, the computation of $\Phi_{A \to B}$ can be performed using a variety of algorithms, as described by Madsen (2006). Evidence is entered in $\mathcal{T}$ by instantiating $\mathcal{X}_\epsilon$ according to ϵ.

4 IMPROVING BELIEF UPDATE

4.1 ARC-REVERSAL SORT

Using AR a variable Y is eliminated by a sequence ρ of arc-reversal operations followed by a barren variable elimination. If $|\mathrm{ch}(Y)| > 1$, then an arc-reversal order $\rho = ((Y, X_1), \ldots, (Y, X_{|\mathrm{ch}(Y)|}))$ has to be determined.

Consider the query $Q = (\mathcal{T} = \{X_1, X_3, X_4, X_5\}, \Phi, \emptyset)$ where $\Phi = \{P(X_1), (P(X_2 \mid X_1), P(X_3 \mid X_2, X_5), P(X_4 \mid X_2), P(X_5)\}$. Eliminating X_2 using AR involves reversing arcs (X_2, X_3) and (X_2, X_4). Figures 2 and 3 show the calculations for the two possible orders $\rho_{\min} = ((X_2, X_4), (X_2, X_5))$ and

[1] $\mathcal{G}^{\mathcal{T}}$ is constructed from the moral graph $\mathcal{G}^{\mathrm{m}}$ of $\mathcal{G}$ by adding undirected edges until the graph is triangulated. A graph is triangulated if every cycle of length greater than three has a chord.

$\rho_{\max} = ((X_2, X_5), (X_2, X_4))$, respectively. The inner circles represent the first arc-reversal operation while the outer circles represent the second arc-reversal operation. Even though the structures of the two graphs are identical, the solutions are different.

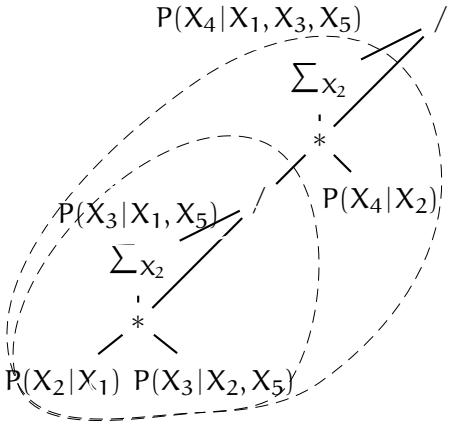


Figure 2: *maximum fill-in-weight.*

The solution illustrated in Figure 2 is $\Phi^{\mathrm{AR\,max}\downarrow \mathsf{T}} = \{\mathsf{P}(X_1), \mathsf{P}(X_3 \mid X_1, X_5), \mathsf{P}(X_4 \mid X_1, X_3, X_5), \mathsf{P}(X_5)\}$ while the solution illustrated in Figure 3 is $\Phi^{\mathrm{AR\,min}\downarrow \mathsf{T}} = \{\mathsf{P}(X_1), \mathsf{P}(X_3 \mid X_1, X_4, X_5), \mathsf{P}(X_4 \mid X_1), \mathsf{P}(X_5)\}$. Notice that the (unique) solution to Q obtained using VE and the algorithm of Section IV in (Madsen, 2006) is $\Phi^{\mathrm{VE}\downarrow \mathsf{T}} = \{\mathsf{P}(X_1), \mathsf{P}(X_3, X_4 | X_1, X_5), \mathsf{P}(X_5)\}$.

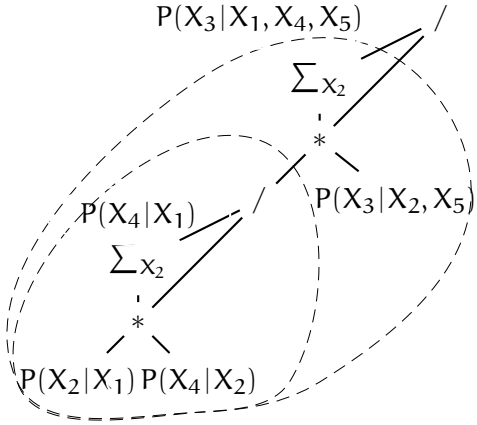


Figure 3: *minimum fill-in-weight.*

The elimination of Y by a sequence of AR operations $\rho = ((Y, X_1), \ldots, (Y, X_{|\mathrm{ch}(Y)|}))$ will induce a set of new edges. The cost[2] of an AR sequence ρ is defined as the sum of the weights of the new edges induced by ρ.

The objective of considering different AR sequences is to minimize the total cost of new edges introduced by eliminating Y. It is infeasible to consider all possible sequences as the upper limit on the number of possible sequences is $n!$ where $n = |\mathrm{ch}(Y)|$. Some of the sequences may be illegal due to the graph acyclicity constraint though. The large number of possible sequences implies that the use of heuristics for determining the sequence to use is justified. We define the cost of reversing edge (Y, X) as:

$$s(Y, X) = \sum_{Z_X \in \mathrm{pa}(X), Z_X \notin \mathrm{pa}(Y), Z_X \neq Y} \|Z_X\| \cdot \|Y\| + \sum_{Z_Y \in \mathrm{pa}(Y), Z_Y \notin \mathrm{pa}(X)} \|Z_Y\| \cdot \|X\|.$$

The cost is equal to the sum of the weights of the edges induced by new parents of X and Y.

We introduce two heuristic rules based on the score $s(Y, X)$: a *minimum fill-in-weight* rule for selecting the next edge to reverse when eliminating Y. We refer to AR in combination with *minimum fill-in-weight* as AR min. The rule where $s(Y, X)$ is maximized is referred to as *maximum fill-in-weight* and AR max denotes AR in combination with this rule.

Both *maximum fill-in-weight* and *minimum fill-in-weight* use a one step look-ahead. This implies that they do not always find the optimal order (according to the cost function). Finding an optimal order is similar to finding an optimal triangulation. It is well-known that this problem is $\mathcal{NP}$-complete, see e.g. (Yannakakis, 1981) or (Arnborg et al., 1987).

4.2 ANY-SPACE

Inspired by the work on RD and RC we extend LP with the any-space property. The basic idea is to avoid computing a representation over all values of ϕ, if $\|\mathrm{dom}(\phi)\| > \delta$ where δ is a threshold value on the size of potentials. Instead of

[2] Alternative scores may be considered. In this work, we use a score similar to the *fill-in-weight* score often used for identifying triangulations using one-step lookahead node elimination, as this rule has shown a high performance when applied to triangulation, see (Kjærulff, 1993) for more details.

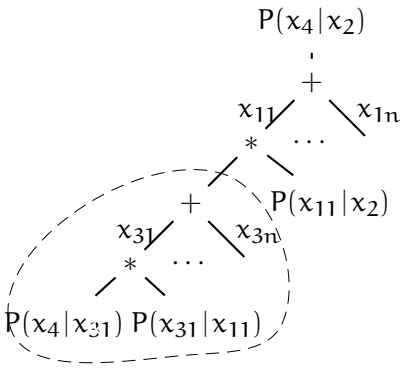


Figure 4: VE calculation of $P(X_4|X_2)$.

maintaining a large (table) representation of ϕ, values are recomputed as needed in subsequent operations. During belief update potential sizes may increase due to multiplications and decrease due to marginalizations. Let ϕ_1 and ϕ_2 be two potentials. If $\mathrm{dom}(\phi_1) \setminus \mathrm{dom}(\phi_2) \neq \emptyset$ or $\mathrm{dom}(\phi_2) \setminus \mathrm{dom}(\phi_1) \neq \emptyset$, then $\|\mathrm{dom}(\phi_1 \cdot \phi_2)\| > \|\mathrm{dom}(\phi_i)\|$ for $i = 1, 2$. This simple insight drives the proposed scheme. The calculation of a product $\prod \phi$ or a marginal $\phi^{\downarrow T}$ is *delayed* if $\|\mathrm{dom}(\prod \phi)\| > \delta$ or $\|\mathrm{dom}(\phi^{\downarrow T})\| > \delta$, respectively. Only marginalization can enforce the construction of a potential whereas both a marginalization and a product may involve delayed potentials producing a recursive scheme. Figure 4 illustrates the approach on:

$$P(X_4|X_2) = \Phi^{\mathrm{VE}\downarrow\{X_2,X_4\}} = \sum_{X_1} P(X_1|X_2) \sum_{X_3} P(X_3|X_1)P(X_4|X_3),$$

where $\Phi = \{P(X_1 \mid X_2), P(X_3 \mid X_1), P(X_4 \mid X_3)\}$. Each entry of $P(X_4|X_2)$ is computed by accessing and combining the values of its source potentials Φ recursively. The equation becomes a formula for accessing the values of $P(X_4|X_2)$ by recursive computation. Each time an entry is accessed, it is computed. No entries are computed when the formula is constructed. This implies that the calculation of an entry is delayed until the entry is accessed as part of the calculation of another potential.

Even though $\|\mathrm{dom}(\phi)\| > \|\mathrm{dom}(\phi^{\downarrow T})\|$, it may be that $\|\mathrm{dom}(\phi^{\downarrow T})\| > \delta$. In this case, the marginalization is postponed. Notice that a marginalization is always performed over a combination of at least two potentials. If $\|\mathrm{dom}(\phi^{\downarrow T})\| \leq \delta$, then $\phi^{\downarrow T}$ is computed.

The results of experiments suggest that VE is the most suited marginalization operation to apply in the any-space scheme. Notice that neither RC nor VU is directly applicable as the marginalization operation in LP.

5 PERFORMANCE EVALUATION

This section presents the results of a preliminary performance evaluation[3]. The evaluation is performed using a set of real-world and randomly generated BNs. The set of real-world networks considered includes *Barley* and *ship-ship* while networks with $\|\mathcal{X}\| = 100, 125, 150, 200$ were generated randomly (ten networks of each size). For each network ten different $\mathcal{X}_\epsilon$ were generated randomly for each $\|\mathcal{X}_\epsilon\| = 0, \ldots, \|\mathcal{X}\|$. Table 1 (where $s(C) = \prod_{X \in C} \|X\|$ and $s(\mathcal{C}) = \sum_{C \in \mathcal{C}} s(C)$) contains statistics on some test networks (in the name *net_x_y* $x = \|\mathcal{X}\|$ and y is an identifier). The junction trees have been generated using *optimal triangulation* (total weight being the optimality criterion) (Jensen, 2007).

Table 1: Statistics on test networks.

Network	$\|V\|$	$\|\mathcal{C}\|$	$\max_{C \in \mathcal{C}} s(C)$	$s(\mathcal{C})$
ship-ship	50	35	4,032,000	24,258,572
Barley	48	36	7,257,600	17,140,796
net_100_5	100	85	98,304	311,593
net_200_5	200	178	15,925,248	70,302,065

This section also presents the results of an evaluation of the cost of the last and most expensive division operation involved in the elimination of a variable by AR. Ndilikilikesha (1994) introduces operations on the DAG structure where the need for division is eliminated. This is achieved by associating a potential instead of a CPD with each variable. This implies that barren variable elimination requires marginalization operations and it therefore becomes a potentially expensive operation.

[3]Due to space restrictions, a limited number of graphs are included for each experiment.

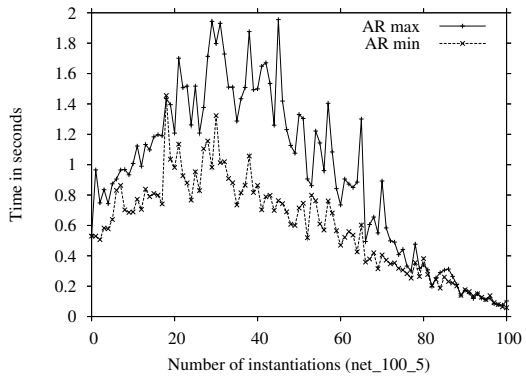


Figure 5: Time cost of LARP with sorting.

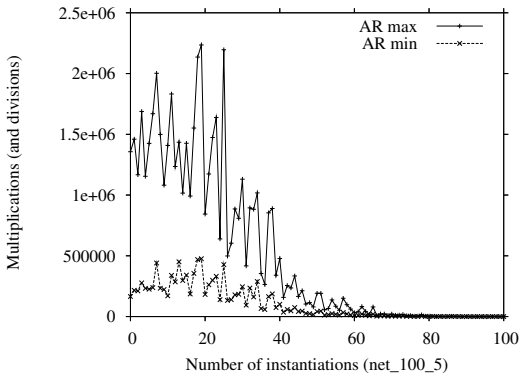


Figure 7: Multiplications & divisions, sorting.

5.1 ARC-REVERSAL SORT

To assess the performance impact of ρ, we compare the costs of belief update using AR min and AR max. The *minimum fill-in-weight* rule selects as the next arc to reverse an arc with lowest cost while the *maximum fill-in-weight* rule selects an arc with highest cost. A performance comparison between AR min and AR max will give insights into the importance of selecting a *good* arc-reversal order.

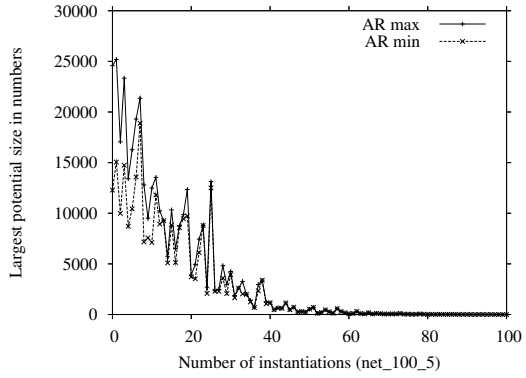


Figure 6: Space cost of LARP with sorting.

Figures 5 and 6 show the cost of belief update in *net_100_5*. The time cost of AR min is significantly lower than the time cost of AR max whereas the reduction in potential size is less significant and it is most significant for small subsets of evidence. Only in a few cases there is a reduction in the largest potential size when using AR min compared to using AR max.

The time cost improvement is not only produced by a reduction in the largest potential size, but also by a reduction in the number of arithmetic operations performed. Figure 7 shows the cost of belief update in *net_100_5* in terms of the number of multiplications and divisions performed. There is a reduction in time cost and number of operations even though there is no reduction in the (average) size of the largest potential.

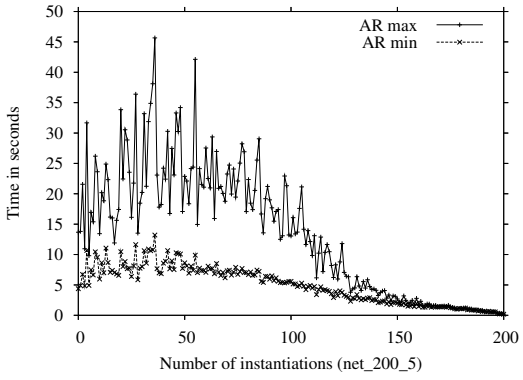


Figure 8: Time cost of LARP with sorting.

Figure 8 shows an example where the use of *minimum fill-in-weight* not only gives a significant reduction in time cost over the use of *maximum fill-in-weight*, but the variation of the cost is also significantly reduced.

We expected the implementation overhead introduced by the sorting algorithm to dominate the time efficiency improvement (e.g., testing for potential cycles in the graph), but this was clearly not the case. The arc-reversal order can have a significant impact on the time cost of belief update. In conclusion, it may be important

to identify an efficient arc-reversal order.

5.2 ANY-SPACE

The any-space property is achieved by not constructing any potential ϕ with $\|\mathrm{dom}(\phi)\| > \delta$. To illustrate the any-space property, we performed a sequence of experiments with different δ values. Notice that reducing δ from $\mathfrak{y}$ to $\mathfrak{x}$ only has an impact on performance when at least one potential ϕ with $\mathfrak{x} < \|\mathrm{dom}(\phi)\| \leq \mathfrak{y}$ is created during belief update.

Figure 9 shows the time cost of belief update in *Barley* for three different δ values ($\max_{S \in \mathcal{S}} s(S) = 907,200$) using VE as the marginalization algorithm. The time cost increases as δ is reduced.

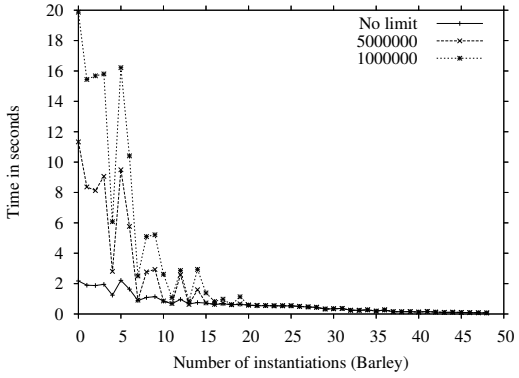


Figure 9: Time cost for different δ values.

The experiments show that the average largest potential size has a major variation. The peaks in the graphs are caused by a few *difficult* evidence scenarios[4]. The combined impact of these scenarios increases as δ decreases.

Table 2 shows the time cost for belief update using AR min and VE in *Barley* given two specific evidence scenarios as a function of δ. The time cost has a large variation across evidence scenarios and the time cost increases as δ decreases. Notice that the time costs for two different values of δ are (almost) equal. The reason is that the largest domain size created during belief update is the same in both cases.

[4]In this case the most expensive set of evidence to propagate consists of four instantiations of leaf variables with multiple parents which are inserted into four different leaf cliques. This evidence introduces additional dependence relations.

Table 2: Time cost of belief update given two different sets of evidence of equal size.

	10^6	$2.5 * 10^6$	$5 * 10^6$	No limit
AR	333.65	41.84	41.50	3.89
VE	18.57	12.35	12.27	2.45
	$5 * 10^3$	$1.5 * 10^4$	$3 * 10^4$	No limit
AR	1.82	1.25	1.25	0.53
VE	2.28	0.96	0.92	0.46

The results of the experiments indicate that the VE algorithm is better suited than AR for implementing upper-limit constraints. The AR algorithm performs additional calculations in order to maintain as many (conditional) independence statements as possible. This seems to penalize the algorithm under upper-limits constraints when compared to VE.

The table indexing for potentials with sizes larger than δ is naïve compared to the table indexing for potentials with sizes below δ. The former table indexing is expected to add an additional overhead to the time costs.

5.3 DIVISION OPERATION

Using AR as the marginalization operation requires one invocation of Equations 1 and 2 for each arc reversed except for the last arc where the invocation of Equation 2 can be skipped as the variable subsequently will be eliminated as barren (Madsen, 2006).

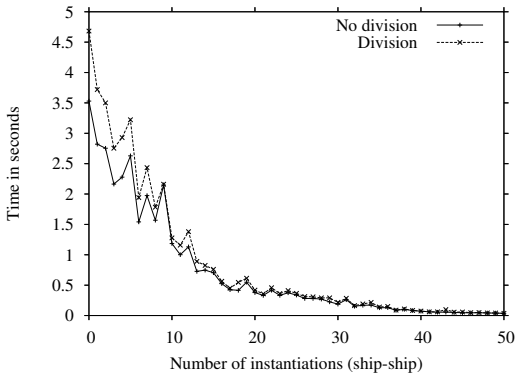


Figure 10: Time cost w/w.o. last division.

Figure 10 illustrates the cost of the division operation in the *ship-ship* network. It is clear from the figure that the cost of the division operation is most significant for the case of small

sized evidence sets. The impact of the division operation is reduced as $\|\mathcal{X}_\epsilon\|$ increases.

6 CONCLUSION

This paper introduces LP as a class of algorithms for computing all-marginals. The elements of the class differ with respect to the algorithm or algorithms used for message computation. We have proposed two methods for message computation in LP and considered the importance of certain properties of the algorithm. One method is based on sorting arc-reversal operations according to a complexity score while the second method is a simple scheme for extending LP with the any-space property. The paper includes an experimental evaluation of the proposed extensions.

Future work includes a more in-depth analysis of the any-space potential of LP in message computation. This includes the option of reconsidering the calculation of a factor at a later point in time. For instance, before accessing the elements of a delayed factor ϕ recursively, it may be possible to identify a more efficient elimination and combination order from the source potentials of ϕ. Future work also includes an analysis of methods for selecting between different algorithms for solving a query. This would produce a propagation scheme where different algorithms may be used to solve different queries during belief update.

References

S. Arnborg, D. G. Corneil, and A. Proskurowski. 1987. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8:277–284.

F. Bacchus, S. Dalmao, and T. Pitassi. 2003. Value Elimination: Bayesian Inference via Backtracking Search. In *Proc. of UAI*, pages 20–28.

C. Cannings, E. A. Thompson, and H. H. Skolnick. 1978. Probability functions on complex pedigrees. *Advances in Applied Probability*, 10:26–61.

G. F. Cooper. 1990a. Bayesian Belief-Network Inference Using Recursive Decomposition. Technical Report KSL 90-05, Knowledge Systems Laboratory.

G. F. Cooper. 1990b. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer-Verlag.

P. Dagum and M. Luby. 1993. Approximating probabilistic inference in Bayesian belief netwoks is NP-hard. *Artificial Intelligence*, 60:141–153.

A. Darwiche and G. Provan. 1997. Query dags: A pratical paradigm for implementing belief-network inference. In *JAIR*, pages 147–176.

A. Darwiche. 2000. Any-Space Probabilistic Inference. In *Proc. of UAI*, pages 133–142.

R. Dechter. 1996a. Bucket elimination: A unifying framework for probabilistic inference. In *Proc. of UAI*, pages 211–219.

R. Dechter. 1996b. Topological Parameters for time-space tradeoff. In *Proc. of UAI*, pages 220–227.

F. V. Jensen and F. Jensen. 1994. Optimal junction Trees. In *Proc. of UAI*, pages 360–366.

F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. 1990. Bayesian Updating in Causal Probabilistic Networks by Local Computations. *Computational Statistics Quarterly*, 4:269–282.

F. Jensen. 2007. *HUGIN API Reference Manual*. Available from http://www.hugin.com.

U. B. Kjærulff and A. L. Madsen. 2008. *Bayesian Networks and Influence Diagrams - A Guide to Construction and Analysis*. Springer-Verlag.

U. B. Kjærulff. 1993. *Aspects of efficiency improvement in Bayesian networks*. Ph.D. thesis, Department of Computer Science, Aalborg University, Denmark, April.

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, B*, 50(2):157–224.

A. L. Madsen and F. V. Jensen. 1999. Lazy propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1-2):203–245.

A. L. Madsen. 2006. Variatoins Over the Message Computation Algorithm of Lazy Propagation. *IEEE Transactions on Systems, Man. and Cybernetics Part B*, 36(3):636–648.

P. Ndilikilikesha. 1994. Potential influence diagrams. *IJAR*, 11(1):251–285.

S. M. Olmsted. 1983. *On representing and solving decision problems*. Phd thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligence Systems*. Series in Representation and Reasoning. Morgan Kaufmann Publishers.

R. D. Shachter, B. D'Ambrosio, and B. Del Favero. 1990. Symbolic probabilistic inference in belief networks. In *Proc. of 8th National Conference on AI*, pages 126–131.

R. D. Shachter. 1986. Evaluating influence diagrams. *Operations Research*, 34(6):871–882.

G. R. Shafer and P. P. Shenoy. 1990. Probability Propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–351.

P. P. Shenoy. 1997. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *IJAR*, 17(2-3):239–263.

M. Yannakakis. 1981. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):77–79.

N. L. Zhang and D. Poole. 1996. Exploiting Causal Independence in Bayesian Network Inference. *Journal of Artificial Intelligence Research*, 5:301–328.

Solving CLQG Influence Diagrams Using Arc-Reversal Operations in a Strong Junction Tree

Anders L Madsen
HUGIN EXPERT A/S, Gasværksvej 5, DK-9000 Aalborg, Denmark
Anders.L.Madsen@hugin.com

Abstract

This paper presents an architecture for solving conditional linear-quadratic Gaussian (CLQG) influence diagrams (IDs) by Lazy Propagation (LP). A strong junction tree (SJT) is used to guide the elimination order whereas the marginalization operation is based on arc-reversal (AR). The use of AR for marginalization simplifies the implementation and gives the architecture a number of advantages. The key benefits of using LP in combination with AR to solve CLQG IDs are illustrated by examples and in experiments. The results of a preliminary performance evaluation are promising.

1 INTRODUCTION

The ID (Howard and Matheson, 1984) is an increasingly popular knowledge representation for decision making under uncertainty. It provides an intuitive graphical representation of a decision problem with a minimum of clutter and confusion for the decision maker and analyst (Shachter and Peot, 1992).

Some of the most popular algorithms for solving IDs, e.g., Olmsted (1983), Shachter (1986), Shenoy (1992), and Jensen et al. (1994), consider the discrete case only. Recently, an increased interest in IDs with continuous as well as mixed continuous and discrete variables has emerged. Kenley (1986) and Shachter and Kenley (1989) introduced the Gaussian ID consisting of continuous variables only and an architecture for its solution. The architecture assumes a single utility function (UF) conditioned on all variables in the ID. The solution process is based on AR operations. Later Poland (1994) introduced an architecture for representing and solving continuous IDs by approximating continuous distributions with Gaussian mixtures. The solution process of this architecture is also based on AR operations. Later Madsen and Jensen (2005) introduced Shenoy-Shafer and LP architectures for solving CLQG IDs. These two architectures are based on a new representation of UFs and representations and operations of Lauritzen and Jensen (2001) and Shachter and Kenley (1989). At the same time, Cobb and Shenoy (2004) introduced an architecture for solving hybrid IDs using mixtures of truncated exponentials (MTEs).

The new architecture introduced in this paper extends LP (Madsen, 2006) with representations for UFs (Madsen and Jensen, 2005) and operations for eliminating random and decision variables (DVs) from UFs. It is simpler than the architecture of Madsen and Jensen (2005). We refer to the proposed architecture as *LARP* as it is based on LP using AR operations (Cowell, 2005; Madsen, 2006) for variable elimination. We make an empirical analysis of the efficiency of LARP using randomly generated IDs and different versions of a famous example. In addition, we compare LARP with the HUGIN algorithm (Jensen et al., 1994).

2 PRELIMINARIES

2.1 CLQG INFLUENCE DIAGRAM

A CLQG ID $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{F}, \mathcal{U})$ over variables $\mathcal{X}$ consists of a DAG $\mathcal{G}$, a set of conditional probability distributions $\mathcal{P} = \{\mathsf{P}(\mathsf{X}|\ \mathsf{pa}(\mathsf{X})) : \mathsf{X} \in \Delta_{\mathsf{C}}\}$ where $\mathsf{pa}(\mathsf{X})$ is the set of variables corresponding to the parents of X in $\mathcal{G}$, a set of con-

ditional linear Gaussian (CLG) density functions $\mathcal{F} = \{f(Y|\,pa(Y)) : Y \in \Gamma_C\}$ and a set of quadratic UFs $\mathcal{U}$ where Δ is the set of discrete variables s.t. Δ_C is the set of discrete random variables (RVs) and Δ_D is the set of discrete DVs, Γ is the set of continuous variables s.t. Γ_C is the set of continuous RVs and Γ_D is the set of continuous DVs, i.e., $\mathcal{X} = \Delta_C \cup \Gamma_C \cup \Delta_D \cup \Gamma_D$. We denote the set of RVs as $\mathcal{X}_C = \Delta_C \cup \Gamma_C$ and the set of DVs as $\mathcal{X}_D = \Delta_D \cup \Gamma_D$ s.t. $\mathcal{X} = \mathcal{X}_C \cup \mathcal{X}_D$. The variables $\Delta_C \cup \Gamma_C$ induce a CLG distribution conditional on $\Delta_D \cup \Gamma_D$ s.t. $P(\Delta_C|\Delta_D) \cdot f(\Gamma_C|\Delta, \Gamma_D)$ equals:

$$\prod_{X \in \Delta_C} P(X|\,pa(X)) \cdot \prod_{Y \in \Gamma_C} f(Y|\,pa(Y)).$$

The variables of $\mathcal{N}$ induce an expected UF:

$$EU(\mathcal{X}) = P(\Delta_C|\Delta_D) \cdot f(\Gamma_C|\Delta, \Gamma_D) \cdot \sum_{u \in \mathcal{U}} u. \quad (1)$$

An optimal strategy can be identified by eliminating variables from (1) in the reverse time order. The time order is the order in which variables are observed s.t. $\mathcal{I}_i$ is the set of variables observed after the ith decision and before the $i+1$th decision.

Let $Y \in \Gamma_C$ with $I = pa(Y) \cap \Delta$ and $Z = pa(Y) \cap \Gamma$, then Y has a CLG distribution if:

$$f(Y|I = i, Z = z) = \mathcal{N}(\alpha(i) + \beta(i)z, \sigma^2(i)), \quad (2)$$

where the mean value is linear in the values of Z, while the covariance matrix is independent of Z. In (2), $\alpha(i)$ is a table of real numbers, $\beta(i)$ is a table of $|Z|$-dimensional row vectors and $\sigma^2(i)$ is a table of non-negative values.

A quadratic UF has the form $u(X = x, I = i) = x^T Q(i) x + R(i) x + S(i)$ where X is a $|X| \times 1$-dimensional vector of continuous variables, $I \subseteq \Delta$, $Q(i) := (q_{jk})_{|X| \times |X|}(i)$ is a table of $|X| \times |X|$ symmetric negative semi-definite matrices, $R(i) = (r_k)_{|X|}(i)$ is a table of $1 \times |X|$ vectors and $S(i)$ is a table of constants. Thus, a UF is represented as a triple $[Q, R, S]$.

We assume the UF to be a negative quadratic function as a weighted average of quadratic functions is quadratic. This implies that optimization of DVs and elimination of RVs from (1) have closed form solutions. Notice that the UFs specified in the model need not be negative quadratic. It is sufficient if the UF over which a continuous DV is maximized is negative quadratic (or constant).

We let $\mathcal{G}(\mathcal{P} \cup \mathcal{F} \cup \mathcal{U})$ denote the domain graph spanned by $\mathcal{P} \cup \mathcal{F} \cup \mathcal{U}$ and $C_{\mathcal{G}}(X)$ denote the conditioning variables of X in $\mathcal{G}$ where subscript $\mathcal{G}$ is omitted when no confusion is possible.

A RV X in $\mathcal{G}(\mathcal{P} \cup \mathcal{F} \cup \mathcal{U})$ is *probabilistic barren* w.r.t. $T \subseteq \mathcal{X}$ if it is barren w.r.t. T in $\mathcal{G}(\mathcal{P} \cup \mathcal{F})$. Let D_i be the ith decision in an ID. A variable $X \in \mathcal{I}_{k>=i}$ is *irrelevant* for D_i if $X \perp de(D_i) \cap U |\, pa(D_i)$ where $\perp$ denotes d-separation and $de(D_i)$ are the descendants of D_i while a variable $Y \in pa(D_i)$ is *non-requisite* for D_i if $X \perp U \cap de(D_i) |\, pa(D_i) \setminus \{Y\}$, see, e.g., Nielsen (2001) for more details.

2.2 THE AR OPERATION

The AR operation (Shachter, 1986; Cowell, 2005) changes the direction of an edge between two variables. Let $X_i, X_j \in \Delta$ with $C(X_i) = \{Z_1, \ldots, Z_n\} \subseteq \Delta$ and $C(X_j) = \{X_i, Z_1, \ldots, Z_n\} \subseteq \Delta$ s.t. $p(X_j|X_i, Z_1, \ldots, Z_n)$ and $p(X_i|Z_1, \ldots, Z_n)$ are the corresponding probability distributions of X_i and X_j, respectively. The edge (X_i, X_j) is reversed by setting:

$$p(X_j|Z_1, \ldots, Z_n) = \sum_{X_i} p(X_j|X_i, Z_1, \ldots, Z_n) p(X_i|Z_1, \ldots, Z_n),$$

$$p(X_i|X_j, Z_1, \ldots, Z_n) = \frac{p(X_j|X_i, Z_1, \ldots, Z_n) p(X_i|Z_1, \ldots, Z_n)}{p(X_j|Z_1, \ldots, Z_n)}. \quad (3)$$

The AR operation can also be applied to a pair of density functions (Cowell, 2005). Let $Y_i, Y_j \in \Gamma$ with $C(Y_j) = \{Z_1, \ldots, Z_n\} \subseteq \Gamma$ and $C(Y_i) = \{Y_j, Z_1, \ldots, Z_n\} \subseteq \Gamma$ s.t.:

$$Y_i|Y_j, Z_1, \ldots, Z_n \sim \mathcal{N}(\alpha_{Y_i} + \beta_{Y_j} Y_j + \sum_{i=1}^{n} \beta_i Z_i, \sigma^2_{Y_i}),$$

$$Y_j|Z_1, \ldots, Z_n \sim \mathcal{N}(\alpha_{Y_j} + \sum_{i=1}^{n} \delta_i Z_i, \sigma^2_{Y_j}).$$

The distributions of Y_i and Y_j after AR are:

$$Y_i|Z_1,\ldots,Z_n \sim$$
$$\mathcal{N}((\alpha_{Y_i}+\beta_{Y_j})+\sum_{i=1}^{n}(\beta_i+\delta_i)Z_i, \sigma^2_{Y_i}+\beta^2_{Y_j}\sigma^2_{Y_j}),$$
$$Y_j|Y_i,Z_1,\ldots,Z_n \sim \mathcal{N}\left(\frac{\alpha_{Y_j}\sigma^2_{Y_i}+\mu}{d}, \frac{\sigma^2_{Y_j}\sigma^2_{Y_i}}{d}\right),$$

where $d = \sigma^2_{Y_i}+\beta^2_{Y_j}\sigma^2_{Y_j}$ and μ equals:

$$-\alpha_{Y_i}\beta_{Y_j}\sigma^2_{Y_j}+\beta_{Y_j}\sigma^2_{Y_j}Y_i+\sum_{i=1}^{n}(\delta_i\sigma^2_{Y_i}-\beta_i\beta_{Y_j}\sigma^2_{Y_j})Z_i.$$

See Cowell (2005) and Madsen (2006) for more details. If $C(Y_i)\cap\Delta = C(Y_j)\cap\Delta = K$, then the formulas are indexed by k.

3 POTENTIALS

Definition 1. A *potential* $\pi_W = (\mathcal{P},\mathcal{F},\mathcal{U})$ on $W \subseteq \mathcal{X}$ consists of a set of conditional probability potentials $\mathcal{P}$ over subsets of W, a set of CLG density functions $\mathcal{F}$ over subsets of $W\cap\Gamma$ conditional on subsets of $W\cap\Delta$ and a set of UFs $\mathcal{U}$ over subsets of W.

Elements of $\mathcal{P}$ are referred to as *factors* and elements of $\mathcal{F}$ as *density functions*. Furthermore, we call the potential $\pi_W = (\emptyset,\emptyset,\emptyset)$ *vacuous* and denote it $\pi_\emptyset$. The domain of $\mathcal{P}$ is $\mathsf{dom}(\mathcal{P}) = \bigcup_{p\in\mathcal{P}}\mathsf{dom}(p)$ where $\mathsf{dom}(p)$ denotes the set of variables over which p is defined (similarly for $\mathsf{dom}(\mathcal{F})$ and $\mathsf{dom}(\mathcal{U})$). The domain of a potential $\pi = (\mathcal{P},\mathcal{F},\mathcal{U})$ is defined as $\mathsf{dom}(\pi) = \mathsf{dom}(\mathcal{P})\cup\mathsf{dom}(\mathcal{F})\cup\mathsf{dom}(\mathcal{U})$. Finally, we define $\mathcal{G}(\pi) = \mathcal{G}(\mathcal{P}\cup\mathcal{F}\cup\mathcal{U})$.

Definition 2. The *combination* of potentials $\pi_{W_1} = (\mathcal{P}_1,\mathcal{F}_1,\mathcal{U}_1)$ and $\pi_{W_2} = (\mathcal{P}_2,\mathcal{F}_2,\mathcal{U}_2)$ denotes the potential on $W_1\cup W_2$ given by $\pi_{W_1}\otimes\pi_{W_2} = (\mathcal{P}_1\cup\mathcal{P}_2,\mathcal{F}_1\cup\mathcal{F}_2,\mathcal{U}_1\cup\mathcal{U}_2)$.

Potential combination is simply set union.

Definition 3. A *projection* of $\pi_W = (\mathcal{P}_W,\mathcal{F}_W,\mathcal{U}_W)$ to a subset $V\subseteq W$ denotes the potential $\pi_V = \pi_W^{\downarrow V} = (\mathcal{P}_V,\mathcal{F}_V,\mathcal{U}_V)$ obtained by performing a sequence of variable eliminations of $W\setminus V$.

In projection $\downarrow$, variables $\Gamma\cap V$ are eliminated before $\Delta\cap V$ in reverse time order.

3.1 MARGINALIZATION

Madsen and Jensen (2005) define the necessary and sufficient marginalization operations for solving a CLQG ID. These operations assume the $[Q,R,S]$-representation of UFs and the $p\cdot[A,B,C]$-representation of (multivariate) CLG distributions where p is the discrete potential and A is a vector of αs, B is a matrix of βs, and C is a covariance matrix (this representation is equal to the $[p,A,B,C]$-representation introduced by Lauritzen and Jensen (2001) except for the decomposition of the potential into the discrete p and the continuous $[A,B,C]$ parts). The main contribution of this paper is that the proposed solution method considers only univariate CLG distributions, i.e., a density function has exactly one head variable. This is a significant simplification over the $[A,B,C]$-representation as it simplifies the operations related to variable elimination and eliminates the need for complex matrix operations.

Madsen (2006) describes an architecture for belief update in CLG Bayesian networks using AR operations to eliminate variables. These operations are applicable for the elimination of RVs in the process of solving a CLQG ID.

When solving an ID, variables are eliminated in the reverse temporal order and continuous variables are eliminated before discrete variables. In the process we may take advantage of irrelevance and non-requisite variables. In the following subsections we assume X is the variable to eliminate, $\mathcal{U}_X = \{u\in\mathcal{U} : X\in\mathsf{dom}(u)\}$ and $\mathsf{dom}(\mathcal{U}_X) = \{X,Z_1,\ldots,Z_n\}$. In the process of eliminating variables it may be necessary to perform straightforward domain extensions.

3.1.1 Discrete Random Variables

The marginalization of $X\in\Delta_C$ from a potential $\pi = (\mathcal{P},\emptyset,\mathcal{U})$ s.t. $\mathsf{dom}(\mathcal{U})\subseteq\Delta$ produces a new potential $\pi^*_{\mathsf{dom}(\pi)-\{X\}} = (\mathcal{P}^*,\emptyset,\mathcal{U}^*)$ where the components $\mathcal{P}^*$ and $\mathcal{U}^*$ are computed as follows. A sequence of AR operations to make X probabilistic barren in $\mathcal{G}(\mathcal{P})$ is performed. Let $\mathcal{P}_X$ denote the resulting set of factors, the set of UFs $\mathcal{U}$ is unchanged by this operation. Since X is probabilistic barren only a single factor $p\in\mathcal{P}_X$ has X in its

domain. Let $p(X|\,C(X))$ be this potential and let $u(X, C(X)) = \sum_{u\in\mathcal{U}_X} u$. Finally, we set:

$$\begin{aligned}\mathcal{P}^* &= \mathcal{P}_X \setminus \{p(X|\,C(X)) \in \mathcal{P}_X\},\\ \mathcal{U}^* &= \mathcal{U} \setminus \mathcal{U}_X \cup \mathcal{U}_{-X},\end{aligned} \tag{4}$$

where $\mathcal{U}_{-X} = \{\sum_X p(X|\,C(X)) \cdot u(X, C(X))\}$ denotes the resulting set of UFs (even though $|\mathcal{U}_{-X}| = 1$). Notice that if $\mathcal{U}_X = \emptyset$, then $\mathcal{U}^* = \mathcal{U}$. Later the marginalization operation is extended to produce a set of UFs.

3.1.2 Continuous Random Variables

The marginalization of $X \in \Gamma_C$ from a potential $\pi = (\mathcal{P}, \mathcal{F}, \mathcal{U})$ produces a new potential $\pi^*_{\mathsf{dom}(\pi)-\{X\}} = (\mathcal{P}, \mathcal{F}^*, \mathcal{U}^*)$ where $\mathcal{F}^*$ and $\mathcal{U}^*$ are computed as follows. A sequence of AR operations to make X probabilistic barren in $\mathcal{G}(\mathcal{P}\cup\mathcal{F})$ is performed. Let $\mathcal{F}_X$ denote the resulting set of density functions, the factors $\mathcal{P}$ and set of UFs $\mathcal{U}$ are unchanged by this operation. Since X is probabilistic barren a single density function $f \in \mathcal{F}_X$ has X in its domain. Let $f(X|\,C(X))$ be this density function and let $u(X, C(X)) = \sum_{u\in\mathcal{U}_X} u$. Finally, we set:

$$\begin{aligned}\mathcal{F}^* &= \mathcal{F}_X \setminus \{f(X|\,C(X)) \in \mathcal{F}_X\},\\ \mathcal{U}^* &= \mathcal{U} \setminus \mathcal{U}_X \cup \mathcal{U}_{-X}.\end{aligned} \tag{5}$$

where $\mathcal{U}_{-X} = \left\{(f(X|\,C(X)) \cdot u(X, C(X)))^{\downarrow C(X)}\right\}$ again denotes the resulting set of UFs (again $|\mathcal{U}_{-X}| = 1$). Notice that if $\mathcal{U}_X = \emptyset$, then $\mathcal{U}^* = \mathcal{U}$. Later the marginalization of $X \in \Gamma_C$ is revised s.t. it may produce a set of UFs.

The operation $(f(X|\,C(X)) \cdot u(X, C(X)))^{\downarrow C(X)}$ produces a UF $u(C(X)) = [(q^*_{ij}), (r^*_i), S^*]$ where:

$$\begin{aligned}q^*_{ij} &= q_{ij} + q_{ik}\beta_j + \beta_i q_{kj} + q_{kk}(\beta_i\beta_j),\\ r^*_i &= r_i + 2\alpha q_{ki} + (2q_{kk}\alpha + r_k)\beta_i,\\ S^* &= S + q_{kk}(\alpha^2 + \sigma^2) + r_k\alpha,\end{aligned}$$

where subscript k specifies the matrix/vector entry corresponding to variable X. If $I = C(X) \cap \Delta \neq \emptyset$, the formulas are indexed by the configuration i of I (similarly for continuous decisions).

3.1.3 Discrete Decision Variables

The marginalization of $X \in \Delta_D$ from a potential $\pi = (\emptyset, \emptyset, \mathcal{U})$ s.t. $\mathsf{dom}(\mathcal{U}) \subseteq \Delta$ produces a new potential $\pi^*_{\mathsf{dom}(\pi)-\{X\}} = (\emptyset, \emptyset, \mathcal{U} \setminus \mathcal{U}_X \cup \{\max_D \sum_{u\in\mathcal{U}_X} u\})$.

An optimal policy δ_X for X is determined as the maximizing arguments of $\sum_{u\in\mathcal{U}_X} u$.

3.1.4 Continuous Decision Variables

The marginalization of $X \in \Gamma_D$ from a potential $\pi = (\emptyset, \emptyset, \mathcal{U})$ s.t. $\mathsf{dom}(\mathcal{U}) \subseteq \Delta \cup \Gamma$ produces a new potential $\pi^*_{\mathsf{dom}(\pi)-\{X\}} = (\emptyset, \emptyset, \mathcal{U}^*)$. $\mathcal{U}^*$ is computed as:

$$\mathcal{U}^* = \mathcal{U} \setminus \mathcal{U}_X \cup \{(\sum_{u\in\mathcal{U}_X} u)^{\downarrow\{Z_1,\ldots,Z_n\}}\},$$

where $(\sum_{u\in\mathcal{U}_X} u)^{\downarrow\{Z_1,\ldots,Z_n\}} = [(q^*_{ij}), (r^*_i), S^*]$,

$$q^*_{ij} = q_{ij} - \frac{q_{ik}q_{kj}}{q_{kk}}, r^*_i = r_i - \frac{r_k q_{ki}}{q_{kk}}, S^* = S - \frac{r_k^2}{4q_{kk}}.$$

An optimal policy $\delta_X(z)$ for X is

$$\delta_X(z_1, \ldots, z_n) = -\frac{r_k + 2\sum_{i=1}^n q_{ki}z_i}{2q_{kk}}.$$

A sufficient condition for the marginalization to be well-defined is $q_{kk} < 0$ (for all discrete configurations with non-zero probability) as this implies that the second order polynomial has a unique maximum with respect to X (Madsen and Jensen, 2005).

4 STRONG JUNCTION TREE

A SJT $\mathcal{T}$ with cliques $\mathcal{C}$ and separators $\mathcal{S}$ is used to solve $\mathcal{N}$. Basically, $\mathcal{T}$ is used as a computational caching structure to guide the solution process, i.e., the order in which variables are eliminated, as $\mathcal{T}$ induce a partial order on the elimination order. $\mathcal{T}$ is constructed by moralization and strong triangulation of $\mathcal{G}$.

4.1 INITIALIZATION

The initialization of $\mathcal{T}$ consists of the following steps: (1) associate $\pi_\emptyset$ with each clique $C \in \mathcal{C}$, (2) for each $X \in \Delta, Y \in \Gamma$, assign $P(X|\,\mathsf{pa}(X)) \in \mathcal{P}, f(Y|\,\mathsf{pa}(Y)) \in \mathcal{F}$ to the clique C closest to strong root R s.t. $\mathrm{fa}(X), \mathrm{fa}(Y) \subseteq C$ where $\mathrm{fa}(X) = \mathsf{pa}(X) \cup \{X\}, \mathrm{fa}(Y) = \mathsf{pa}(Y) \cup \{Y\}$ and (3) for each UF $u \in \mathcal{U}$, assign u to the clique C closest to R s.t. $\mathsf{dom}(u) \subseteq C$. After initialization each

clique C holds a potential $\pi_C = (\mathcal{P}, \mathcal{F}, \mathcal{U})$. The potential $\pi_{\mathcal{X}} = \bigotimes_{C \in \mathcal{C}} \pi_C$ on $\mathcal{T}$ is a decomposition of the expected UF over $\mathcal{X}$:

$$\Big(\bigcup_{X \in \Delta_C} \{P(X|\, \mathsf{pa}(X))\}, \bigcup_{Y \in \Gamma_C} \{f(Y|\, \mathsf{pa}(Y))\}, \bigcup_{u \in \mathcal{U}} \{u\} \Big).$$

4.2 MESSAGE PASSING

The solution process in $\mathcal{T}$ proceeds by message passing via the separators $\mathcal{S}$. The separator $S = A \cap B$ between two adjacent cliques A and B stores the message passed between A and B. Messages are passed from leaf cliques toward R by recursively letting each clique A pass a message to its parent B whenever A has received a message from each $C \in \mathrm{adj}(A) \setminus \{B\}$ where $\mathrm{adj}(C)$ is the set of cliques adjacent to C.

Hence, a clique A sends a message $\pi_{A \to B}$ to its parent B when it has received messages from all its children s.t. $\pi_{A \to B} = \left(\pi_A \otimes \left(\otimes_{C \in \mathrm{adj}(A) \setminus \{B\}} \pi_{C \to A}\right)\right)^{\downarrow B}$ where $\pi_{C \to A}$ is the message passed from C to A.

5 SOLVING A CLQG ID

The solution of a CLQG ID $\mathcal{N}$ using LARP proceeds in two steps: 1) construction and initialization of a SJT representation $\mathcal{T}$ of $\mathcal{N}$ and 2) a round of message passing from the leaves of $\mathcal{T}$ to its root R. In the process of eliminating a DV X from a UF u, an optimal policy δ_X for X is recorded as the maximizing arguments of u as described in Sections 3.1.3 and 3.1.4.

Once the variables of $\mathcal{X}$ have been eliminated, an optimal strategy $\widehat{\Delta} = \{\delta_X : X \in \mathcal{X}_D\}$ has been identified and $\mathsf{EU}(\widehat{\Delta})$ computed.

Since an ID over discrete variables only is a special case of a CLQG ID, LARP can also be used to solve discrete IDs.

5.1 DISTRIBUTIVE LAW

The distributive law of algebra (DL) may be exploited in the solution process, see e.g., (Madsen and Jensen, 1999; Dechter, 2000; Pralet et al., 2006). Consider the example where $X \in \Delta_C$ is to be eliminated from a sum of UFs:

$$U(Y, T, Z) = \sum_X P(X) \left(U(X, Y, Z) + U(X, T)\right).$$

Using DL the expression is rewritten:

$$U(Y, Z) + U(T) = \sum_X P(X) U(X, Y, Z) + \sum_X P(X) U(X, T).$$

This use of DL reduces the number of arithmetic operations in the marginalization of X and supports the decompositions of a UF into a set of UFs. This may reduce the total number of arithmetic operations. In (4) and (5) $\mathcal{U}_{-X}$ is specified as a set of UFs to indicate a possible use of DL. Figure 1 shows an example where the application of DL reduces the computational cost significantly.

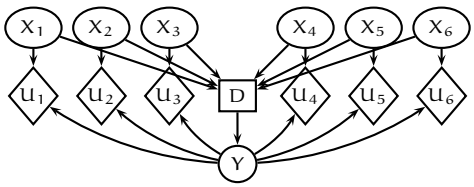


Figure 1: Example of DL utilization.

If we assume $\|Y\| = 100$, $\|X_i\| = 5$ and $\|D\| = 10$, then the advantage of applying DL becomes apparent. The optimal SJT for $\mathcal{N}$ consists of a single clique including all variables. Hence, there is no structure to exploit in the graph $\mathcal{G}$. The state space size of the largest factor when DL is not used is $15,626,000$ whereas it is $156,250$ when DL is used. In the former case $\mathcal{U}_{-X} = \{U(X_1, \ldots, X_6, D)\}$ and in the latter case $\mathcal{U}_{-X} = \{U(X_1, D), \ldots, U(X_6, D)\}$.

The SPI principle (Shachter et al., 1990) may be applied to combine UFs pairwise. The variable X may be eliminated at any step of the process, i.e., X may be eliminated from each UF separately, from the sum of a subset of UFs or from the sum of all UFs. The latter is the traditional approach. Notice that the result of eliminating X is a set of UFs with zero to n UFs where $n = |\mathcal{U}_X|$ (zero only when $n = 0$).

5.2 DECOMPOSITION

Decomposition of clique and separator potentials facilitates, for instance, the exploitation of irrelevant variables in the solution process. Figure 2 shows the *jjd* network (Jensen et al., 1994)

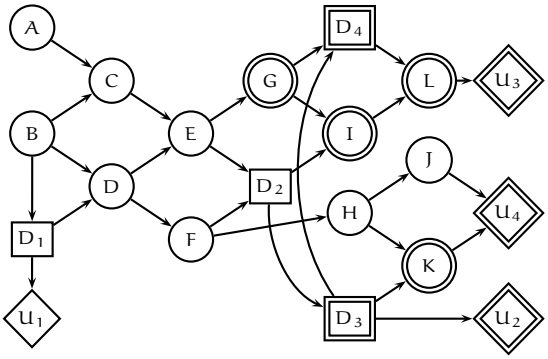


Figure 2: The *jjd* network with mixed variables.

and Figure 3 shows a SJT representation (separators are not shown while link directions specify the flow of messages towards the strong root during the solution process).

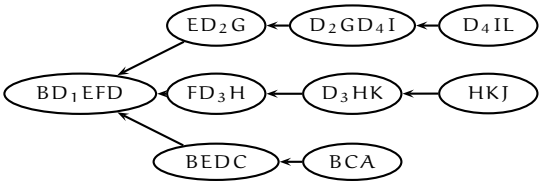


Figure 3: A SJT for *jjd*.

From Figure 2 and the definition of relevance, it is clear that RV D is irrelevant for decisions D_2, D_3 and D_4. In the root clique the elimination process may proceed as:

$$\begin{aligned} \mathrm{EU}(\hat{\Delta}) &= \sum_{B} P(B) \max_{D_1} \big(U_1(D_1) \\ &\quad + \sum_{D} P(D|B,D_1) \big(\sum_{E} P(E|D) U(E) \\ &\quad + \sum_{F} P(F|D) U(F) \big) \big) . \end{aligned}$$

This more efficient calculation violates the strong elimination order, but it is facilitated by the decomposition which enables the algorithm to exploit the irrelevance of D, Figure 4 shows $\mathcal{G}(\pi_{BD_1EFD})$.

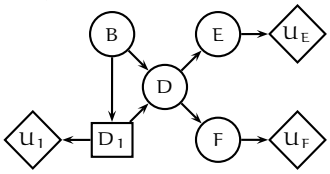


Figure 4: Domain graph for π_{BD_1EFD}.

6 COMPARISON

Due to space constraints comparisons are brief.

6.1 JENSEN ET AL

The HUGIN architecture solves an ID by message passing in a SJT representation. The absorption algorithm is derived based on a variable elimination approach. The initialization process combines all probability distributions assigned to a clique to form the initial clique probability potential. Similarly, for the initial utility potential. Messages are passed from the leaf cliques towards the strong root. Since an ID is solved by a collect operation from the leaves to the root, the HUGIN architecture extended to CLQG IDs reduces to a scheme, which in principle is equivalent to the Shenoy-Shafer architecture (Madsen and Jensen, 2005).

6.2 MADSEN AND JENSEN

Madsen and Jensen (2005) describe Shenoy-Shafer and LP schemes for solving a CLQG ID by message passing in a SJT. The two architectures are based on the $[p, A, B, C]$ CG potential and the $p \cdot [A, B, C]$ CG potential representation, respectively. Madsen and Jensen (2005) assume that appropriate conditioning operations have been performed as a preprocessing step before eliminating a continuous RV as CG potentials, in general, may have multiple head variables. This may require complex matrix operations. In LARP each regression is uni-variate due to the use of the AR operation. This eliminates the need for complicated combination, conditioning and matrix operations and it simplifies the implementation of the architecture.

6.3 COBB AND SHENOY

Since the class of CLQG IDs is quite restricted, MTEs have been considered for solving hybrid IDs (Cobb and Shenoy, 2004; Cobb and Shenoy, 2007) using approximation. The use of MTEs implies that there are no constraints on the relation between $\mathcal{X}_\Gamma$ and $\mathcal{X}_\Delta$ as long as the distributions of the model can be approximated using MTEs. The approach requires a multiplicative decomposition of the UF and $|\mathsf{pa}(X)| \leq 1$

for $X \in \Gamma_D$. These are important limitations. Also, the numerical stability and the number of terms in mixtures are major concerns.

7 PERFORMANCE ANALYSIS

7.1 RANDOM NETWORKS

We compare the performance of LARP and the HUGIN algorithm (Jensen et al., 1994) as implemented in the HUGIN Decision Engine (HDE) on a set of randomly generated networks. The random networks — generated using a revised version of the algorithm used by Vomlelová (2003) — are all discrete as the HDE does not support CLQG IDs. We include this performance comparison as LARP can be used to solve discrete IDs. Table 1[1] shows the results of experiments on eight selected networks. Networks where both algorithms ran out of memory and networks that were solved in less than a few milliseconds were disregarded. In the table, an *N/A* specifies that the solution algorithm ran out of memory. The results indicate

Table 1: Statistics on random networks.

	Time		Space	
$\|\mathcal{X}\|$	LARP	HDE	LARP	HDE
20	4.27	N/A	1,953,125	N/A
20	0.93	1.25	390,625	1,953,125
20	0.03	0.24	3,125	390,625
25	0.13	N/A	15,625	N/A
25	0.64	1.74	78,125	1,953,125
50	4.67	10.16	1,048,576	8,388,608
50	24.31	N/A	4,194,304	N/A
50	7.22	28.64	1,048,576	16,777,216

that LARP is most efficient on the networks solved by both algorithms and that it is able to solve more complex networks. The HDE performs both a collect and a distribute on the SJT though, but it has more efficient data structures and operations. LARP achieves its efficiency by maintaining decompositions of potentials.

7.2 MIXED NETWORKS

Since the HDE does not support CLQG IDs (in fact we are not aware of any system implementing CLQG IDs), we assess the performance of LARP by solving a set of CLQG IDs with the same structure, but different fractions of continuous and discrete variables. For the case of discrete variables only, the network is solved with both LARP and HDE. This will give a rough estimate on the performance of LARP.

Table 2 shows statistics on the IDs *jjd_d*, *jjd_m* and *jjd_c* where $s(C) = \prod_{X \in \Delta \cap C} \|X\|$ and $s(\mathcal{C}) = \sum_{C \in \mathcal{C}} s(C)$. The structure of the network is shown in Figure 2 where $\|X\| = 25$ for $X \in \Delta$. *jjd_d* has discrete variables only, *jjd_m* has mixed variables as indicated in Figure 2 and *jjd_c* has continuous variables only. In Table 3 the average time cost of the solution process is shown for each network. Only for *jjd_d* a value is specified for HDE. The performance of the LARP on the *jjd_d* network is a factor of seven better than the performance of HDE. The performance of LARP improves as the fraction of continuous variables increases. Each network is solved 25 times and the average time cost is computed.

Table 2: Statistics on *jjd_d*, *jjd_m* and *jjd_c*.

jjd	$\|\mathcal{C}\|$	$\max_{C \in \mathcal{C}} s(C)$	$s(\mathcal{C})$
d	9	9,765,625	10,640,625
m	9	9,765,625	10,188,826
c	9	1	1

Table 3: Average time costs in seconds.

	Time		Space	
jjd	HDE	LARP	HDE	LARP
d	3.87	0.53	9,765,625	390,625
m	-	0.35	-	390,625
c	-	0.03	-	1

7.3 DISTRIBUTE LAW

To illustrate the potential impact of DL during the solution process we solved the ID of Figure 1 using HDE, LARP and LARP with DL. The average time costs in seconds (over ten runs) are 2.91, 16.73, and 0.49, respectively. Exploiting DL produced an improvement of a factor of 34 in the time cost when $\|Y\| = 100$, $\|X_i\| = 5$ and $\|D\| = 10$. This (naïve) example was designed to illustrate the potential improvement offered by exploiting DL in the solution process.

[1] Networks with $|\mathcal{X}| \leq 25$ has $\|X\| = 5$ while networks with $|\mathcal{X}| = 50$ has $\|X\| = 2$

A simple heuristic is used to guide the application of DL on UFs. The rule is based on the score $s_{\mathrm{DL}}(\mathsf{p}_Y, \mathcal{U}) = \sum_{\mathsf{u} \in \mathcal{U}_Y} \| \mathsf{dom}(\mathsf{p}_Y) \cup \mathsf{dom}(\mathsf{u}) \|$. If the sum of the state spaces is less than the total state space, i.e., $s_{\mathrm{DL}}(\mathsf{p}_Y, \mathcal{U}) < \| \mathsf{dom}(\mathsf{p}_Y) \cup \mathsf{dom}(\mathcal{U}) \|$, then DL is applied.

8 DISCUSSION

The two main reasons for considering AR operations in a SJT are: 1) the structure of the SJT serves as an efficient caching structure where elimination orders are reconsidered dynamically and 2) the SJT offers an opportunity to distribute information. The first point is relevant if the decision problem is solved on-line (for instance, if unexpected observations have become available or the model is too large to be solved off-line) while the second point is relevant for computing probabilities of future decisions, i.e., the decision policies are encoded as conditional probability distributions, which makes it possible to compute probabilities of future decisions and events under the encoded strategy (Nilsson and Jensen, 1998).

If the UFs of the model meet the requirements of CLQG IDs, then expectation and maximization calculations have closed form solutions. This implies that the ID can be solved. In general, a CLQG IDs can be solved in closed form when all continuous variables can be eliminated before discrete variables using closed form operations. Shenoy (2006) describes an approach to modeling hybrid Bayesian networks using mixtures of Gaussian distributions. This approach can also be applied to approximate continuous distributions in mixed IDs. The resulting mixed ID should meet the requirements of a CLQG ID in order to be solvable with LARP.

Despite a significant difference in the efficiency of table operations LARP is as efficient as the HDE. The results of the performance evaluation indicate a large potential of LARP.

References

B. R. Cobb and P. P. Shenoy. 2004. Decision Making with Hybrid Influence Diagrams Using Mixtures of Truncated Exponentials. In *Proc. of UAI*, pages 85–93.

B. R. Cobb and P. P. Shenoy. 2007. Influence Diagrams with Continuous Decisions Variables and Non-Gaussian Uncertainties. *Decision Analysis*, pages 136–155.

R. G.. Cowell. 2005. Local Propagation In Conditional Gaussian Bayesian Networks. *Journal of Machine Learning Research*, 6:1517–1550.

R. Dechter. 2000. A New Perspective on Algorithms for Optimizing Policies under Uncertainty. In *Artificial Intelligence Planning Systems*, pages 72–81.

R. A. Howard and J. E. Matheson. 1984. Influence Diagrams. In *Readings in Decision Analysis*, chapter 38, pages 763–771.

F. Jensen, F. V. Jensen, and S. L. Dittmer. 1994. From Influence Diagrams to Junction Trees. In *Proc. of UAI*, pages 367–373.

C. R. Kenley. 1986. *Influence Diagram Models with Continuous Variables*. Ph.D. thesis, EES Department, Stanford University.

S. L. Lauritzen and F. Jensen. 2001. Stable Local Computation with Mixed Gaussian Distributions. *Statistics and Computing*, 11(2):191–203.

A. L. Madsen and F. V. Jensen. 1999. Lazy Evaluation of Symmetric Bayesian Decision Problems. In *Proc. of UAI*, pages 382–390.

A.L. Madsen and F. Jensen. 2005. Solving linear-quadratic conditional Gaussian influence diagrams. *International Journal of Approximate Reasoning*, 38(3):263–282.

A. L. Madsen. 2006. Variatoins Over the Message Computation Algorithm of Lazy Propagation. *IEEE Transactions on Systems, Man. and Cybernetics Part B*, 36(3):636–648.

T. D. Nielsen. 2001. Decomposition of Influence Diagrams. In *Proc. of ECSQARU*, pages 144–155.

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 (IPMU)*, pages 144–1462.

S.M. Olmsted. 1983. *On representing and solving decision problems*. Phd thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.

W. B. Poland. 1994. *Decision Analysis with Continuous and Discrete Variables: A Mixture Distribution Approach*. Ph.D. thesis, Engineering-Economic Systems, Stanford University, Stanford, CA.

C. Pralet, T. Schiex, and G. Verfaillie. 2006. From Influence Diagrams to Multioperator Cluster DAGs. In *Proc. of UAI*, pages –.

R. D. Shachter and C. R. Kenley. 1989. Gaussian influence diagrams. *Management Science*, 35(5):527–549.

R. D. Shachter and M. A. Peot. 1992. Decision making using probabilistic inference methods. In *Proc. of UAI*, pages 276–283.

R. D. Shachter, B. D'Ambrosio, and B. Del Favero. 1990. Symbolic probabilistic inference in belief networks. In *Proc. of 8th National Conference on AI*, pages 126–131.

R. D. Shachter. 1986. Evaluating influence diagrams. *Operations Research*, 34(6):871–882.

P. P. Shenoy. 1992. Valuation-Based Systems for Bayesian Decision Analysis. *Operations Research*, 40(3):463–484.

P. P. Shenoy. 2006. Inference in Hybrid Bayesian Networks Using Mixtures of Gaussians. In *Proc. of UAI*, pages 428–436.

M. Vomlelová. 2003. Unconstrained Influence Diagrams - Experiments and Heuristics. T. R, Faculty of Mathematics, Charles University.

Computing the Multinomial Stochastic Complexity in Sub-Linear Time

Tommi Mononen and Petri Myllymäki
Helsinki Institute for Information Technology (HIIT), Finland
{firstname}.{lastname}@hiit.fi

Abstract

Stochastic complexity is an objective, information-theoretic criterion for model selection. In this paper we study the stochastic complexity of multinomial variables, which forms an important building block for learning probabilistic graphical models in the discrete data setting. The fastest existing algorithms for computing the multinomial stochastic complexity have the time complexity of $\mathcal{O}(n)$, where n is the number of data points, but in this paper we derive sub-linear time algorithms for this task using a finite precision approach. The main idea here is that in practice we do not need exact numbers, but finite floating-point precision is sufficient for typical statistical applications of stochastic complexity. We prove that if we use only finite precision (e.g. double precision) and precomputed sufficient statistics, we can in fact do the computations in sub-linear time with respect to data size and have the overall time complexity of $\mathcal{O}(\sqrt{dn} + L)$, where d is precision in digits and L is the number of values of the multinomial variable. We present two fast algorithms based on our results and discuss how these results can be exploited in the task of learning the structure of a probabilistic graphical model.

1 Introduction

Stochastic complexity (SC) is an information-theoretic model selection criterion, which can be seen as a theoretical instantiation of the minimum description length (MDL) principle (Rissanen, 2007; Grünwald, 2007). Intuitively speaking, the basic idea is that the best model for the data is the one which results in the shortest description for the data together with the model. This principle gives us a non-informative, objective criterion for model selection, but there are many ways to define the stochastic complexity formally; one theoretically solid way is to use the *normalized maximum likelihood* (NML) distribution. Recent results suggest that this criterion performs very well in the task of learning Bayesian network structures (Roos et al., 2008).

In the following, let $\mathcal{M}$ denote a parametric probabilistic model, and $\hat{\theta}(\mathbf{x}^n)$ the maximum likelihood parameters of the model given a matrix of observations $\mathbf{x}^n$. The NML distribution is defined as

$$P_{NML}(\mathbf{x}^n \mid \mathcal{M}) = \frac{P(\mathbf{x}^n \mid \hat{\theta}(\mathbf{x}^n), \mathcal{M})}{\sum_{\mathbf{y}^n} P(\mathbf{y}^n \mid \hat{\theta}(\mathbf{y}^n), \mathcal{M})}, \quad (1)$$

where in the numerator we have the maximum likelihood of our observed data and in the denominator we have the sum of maximum likelihoods (denoted in the sequel by $\mathcal{C}(\mathcal{M}, n)$) over all the discrete data sets of size n (Shtarkov, 1987).

Let us define the stochastic complexity as the negative logarithm of (1):

$$SC(\mathbf{x}^n \mid \mathcal{M}) = -\log \frac{P(\mathbf{x}^n \mid \hat{\theta}(\mathbf{x}^n), \mathcal{M})}{\mathcal{C}(\mathcal{M}, n)}. \quad (2)$$

The basic model selection task is to compute the value of this model selection criterion for parametric models of different complexity and choose the one for which this value is minimized, given the observed data.

The single multinomial variable model is an important building block for building more complex probabilistic graphical models for discrete data. For this reason we want to able to compute the NML for multinomial variables as efficiently as possible. In the following, we simplify our notation and leave out $\mathcal{M}$: the model is implicitly defined by the number of values of the multinomial variable, denoted

by L. The numerator is now

$$P(x^n \mid \hat{\theta}(x^n), L) = \prod_{k=1}^{L} \left(\frac{h_k}{n}\right)^{h_k}, \qquad (3)$$

where h_k is a number of data points assigned to the kth value. We expect in this paper that sufficient statistics is known and computing (3) takes therefore only time $\mathcal{O}(L)$. The denominator is

$$\mathcal{C}(L,n) = \sum_{h_1+\cdots+h_L=n} \frac{n!}{h_1!\cdots h_L!} \prod_{k=1}^{L} \left(\frac{h_k}{n}\right)^{h_k},$$

which is a sum of maximum likelihoods so that the summation goes over every possible data of length n.

Although using the definition directly for computing the *multinomial normalizing sum* in the denominator is not computationally feasible, several algorithms for doing this in $\mathcal{O}(n)$ time have been recently developed (Kontkanen and Myllymäki, 2007; Mononen and Myllymäki, 2008b). In this paper we show that our earlier theoretical results presented in (Mononen and Myllymäki, 2008b) can be used for constructing algorithms that compute $\mathcal{C}(L,n)$ in sub-linear time with any desired (finite) precision. We start by briefly reviewing the relevant earlier results and then show how they can be exploited in deriving new ultra-fast algorithms.

2 Known Properties of the Normalizing Sums

In Mononen and Myllymäki (2008b) we proved that the multinomial normalizing sum can be described as a confluent hypergeometric function evaluated at a certain point. We showed that we can write the hypergeometric presentation also in another simple form using falling and rising factorial polynomials. The *falling factorial polynomials* are of the form

$$x^{\underline{k}} = x(x-1)\cdots(x-k+1), \qquad (4)$$

and the *rising factorial polynomials* are

$$x^{\overline{k}} = x(x+1)\cdots(x+k-1). \qquad (5)$$

The *binomial normalizing sum* is then

$$\mathcal{C}(2,n) = \sum_{k=0}^{n} b_k = \sum_{k=0}^{n} \frac{n^{\underline{k}}}{n^k}, \qquad (6)$$

and the general multinomial normalizing sum can be written as

$$\mathcal{C}(L,n) = \sum_{k=0}^{n} m_k = \sum_{k=0}^{n} \frac{n^{\underline{k}}\,(L-1)^{\overline{k}}}{n^k k!}. \qquad (7)$$

As these forms spin off from hypergeometric forms, and a hypergeometric series has the property that there exist a simple ratio of consecutive terms, we know that also (6) and (7) have this property. We will introduce these ratios later in sections 3.1 and 4.1 and use them in the computations.

It is known that the binomial normalizing sum equals to the expectation of the *birthday problem* with the mapping: data size is equal to the number of days (Mononen and Myllymäki, 2008b). We will use an approximation derived for this expectation later in our proof.

There is a recurrence formula for computing the multinomial normalizing sum as the value of the corresponding binomial normalizing sum is known (Kontkanen and Myllymäki, 2007):

$$\mathcal{C}(L,n) = \mathcal{C}(L-1,n) + \frac{n \cdot \mathcal{C}(L-2,n)}{L-2}, \qquad (8)$$

and $\mathcal{C}(1,n)$ is defined to be 1 for every n. This formula can be effectively used for linear time computation of multinomial normalizing sums.

3 The Binomial Normalizing Sum

3.1 Properties of the Sum Terms

Let us start by plotting the terms of (6). We can immediately observe that the first terms of the sum give the greatest impact and most of the terms are very small (see Figure 1). All the terms are positive and getting closer to the zero, because the ratio of successive terms of (6) is

$$\frac{b_k}{b_{k-1}} = \frac{n-k+1}{n}. \qquad (9)$$

Now the natural question is, how many terms do we need, if we want to compute the normalizing sum and use for example double precision. We study this question more closely in section 3.2, but in loose terms the number of needed terms is proportional to the square root on n, which promises sub-linear time performance.

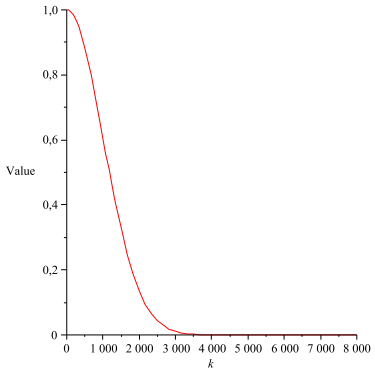


Figure 1: Magnitude of the first 8000 terms of the binomial normalizing sum when the data size (n) is one million.

However, first we have to quantify how to measure the precision. We measure in the standard way the error we make when pruning the sum: we compute the tail sum and compare it to the whole sum. Thus we compute a relative error. However, because setting the desired relative error directly is quite cumbersome, we rather compute it in digits: it much easier just to say that we need e.g. 7 digit precision. More precisely, we define that for positive real numbers p and q, where $p \leq q$ and $p, q \geq 1$, the precision in digits is

$$\left\lfloor -\log_{10}\left(\frac{q-p}{q}\right)\right\rfloor. \tag{10}$$

So if the target q is for example 1.100000 and the approximation p is 1.099999, the precision is $\lfloor 6.04139 \rfloor$. So although only the first digit is the same, the precision in digits according to the definition above is 6. But if we round p after the sixth digit, we get 1.10000. This means that although the precision in digits does not tell all the time how many correct digits we have, it is still very close to what we want. Next we do an upper bound approximation of the last required term of the binomial normalizing sum for achieving some fixed precision.

3.2 Proof of the Right Bound

There is no known closed form solution for (6). To compute the precision, we have to compute the sum starting from some b_r to all the way to the the term b_n. As said before, we are interested in this tail sum, because it tells us how big an error we make, if we stop computing the sum after the term b_{r-1}. Although the terms of the sum look simple, they are a bit tricky to handle, and therefore we perform several upper bound approximations for the terms. Upper bound approximations are of course required, because we want to be sure that whatever bound we get, it must give the promised result. We also move from the discrete sum to an integral presentation, because it makes things simpler in this case.

Next we give in a row four propositions needed for proving the index bound of the binomial sum. After the propositions we prove the mentioned bound, which we call the *right index bound* (in the multinomial case also the left bound exists).

Proposition 1. *We have the following upper bound approximation for the term b_k:*

$$\frac{n^{\underline{k}}}{n^k} \leq \left(1 - \frac{k-1}{2n}\right)^{k-1}, \textit{where} \quad 1 \leq k \leq n.$$

Proof. We prove that the ratio of both sides is bigger than one. Let us look at the ratio:

$$\frac{\left(1-\frac{k-1}{2n}\right)^{k-1}}{\frac{n^{\underline{k}}}{n^k}} = \frac{n \cdot n^{k-1}\left(\frac{n-\frac{1}{2}(k-1)}{n}\right)^{k-1}}{n^{\underline{k}}} \tag{11}$$

$$= \frac{(n-\frac{1}{2}(k-1))^{k-1}}{(n-1)\cdots(n-k+1)} \tag{12}$$

$$= \prod_{r=2}^{k} \frac{n-\frac{1}{2}(k-1)}{n-r+1} = \prod_{r=2}^{k} a_r. \tag{13}$$

Now we just look at the product of pairwise terms defined by

$$a_r a_{k-r+2} = \frac{(n-\frac{1}{2}(k-1))^2}{(n-r+1)(n-k+r-1)} = \frac{N_r}{D_r}.$$

We want to prove that each of these pairwise terms is bigger than 1. However, we can equivalently subtract the denominator from the numerator and require the result to be bigger than 0, because both are always positive in our range. The result is

$$N_r - D_r = n + \frac{1}{4}k^2 - rk + \frac{1}{2}k + r^2 - 2r + \frac{5}{4}. \tag{14}$$

We take the derivate of this difference with respect to k and get the minimum point $k = 2r - 1$. Substituting this in (14), we notice that the result is

$n-r+1$, which is always bigger than 0 in our range. This means that also the pairwise terms must be always bigger than 1, which implies that the proposition must be true for even number of terms a_r.

If we have an odd number of terms a_r, then we have to still prove that the median term is also bigger than 1. The median term is

$$a_{\frac{k}{2}+1} = \frac{2n-k+1}{2n-k}. \tag{15}$$

This cannot be smaller than 1, because $n \geq k \geq 0$. □

Proposition 2. *The following inequality is true for* $\frac{k}{2n} < 1$*:*

$$\left(1 - \frac{k}{2n}\right)^k \leq e^{-\frac{k^2}{2n}}.$$

Proof. Take the natural logarithm of both sides of the inequality and use the known logarithm inequality $\ln(1+x) \leq x$, which is valid for all $x > -1$. □

Proposition 3. *Because* b_k *is a continuous monotonically decreasing function inside the interval* $[0, n]$*, the discrete volume (the sum) corresponds to the upper Riemann sum with intervals of length one. Hence we can give the following inequality:*

$$\sum_{k=0}^{n} b_k \leq 1 + \int_{k=1}^{n} b_{k-1} \mathrm{d}k.$$

Proof. Our integral is always bigger than the given upper sum, because on the right hand side we shifted our function in such way that every discrete column is always entirely below the curve. The size of the first column is 1, which is the first term on the right hand side. □

Proposition 4. *The following inequality holds:*

$$\text{erf}\,(x) \geq \sqrt{1 - e^{-x^2}},$$

when $x \geq 0$ *and*

$$\text{erf}\,(x) = \frac{2}{\sqrt{\pi}} \int_{t=0}^{x} e^{-t^2} \mathrm{d}t.$$

Proof. First suppose $x \geq 0$. Let us now modify the inequality:

$$\text{erf}\,(x) \geq \sqrt{1 - e^{-x^2}} \tag{16}$$

$$(\text{erf}\,(x))^2 + e^{-x^2} - 1 \geq 0 \tag{17}$$

Denote the left hand side by $h(x)$ and take the derivative of it:

$$h'(x) = \frac{2e^{-x^2}(2\text{erf}\,(x) - x\sqrt{\pi})}{\sqrt{\pi}}. \tag{18}$$

We can see that all the roots in our range are solutions for the equation

$$2\text{erf}\,(x) - x\sqrt{\pi} = 0, \tag{19}$$

$$\text{erf}\,(x) = \frac{\sqrt{\pi}}{2} x. \tag{20}$$

As the error function is a convex function between 0 and infinity and the right hand side of (20) is a linear function, there can be only two solutions. The first one is $x = 0$, where $h(0) = 0$, and the other one is $x \approx 0.8982$. The second solution is a positive maximum point. As we have

$$\lim_{x \to \infty} h(x) = (1)^2 + 0 - 1 = 0, \tag{21}$$

this means that $h(x)$ must be always positive in the range, which completes our proof. □

Finally we are now ready to introduce our main theorem giving the right bound approximation:

Theorem 1. *Given precision in digits* (d) *and data size* n*, the right index bound* t *for the binomial normalizing sums is* $\left\lceil 2 + \sqrt{-2n \ln(2 \cdot 10^{-d} - 100^{-d})} \right\rceil$.

Proof. First we approximate the upper bound of the partial binomial normalizing sum from 0 to r:

$$\sum_{k=0}^{r} \frac{n^{\underline{k}}}{n^k} \leq 1 + \sum_{k=1}^{r} \left(1 - \frac{k-1}{2n}\right)^{k-1} \tag{22}$$

$$\leq 1 + \sum_{k=1}^{r} e^{-\frac{(k-1)^2}{2n}} \tag{23}$$

$$\leq 2 + \int_{k=2}^{r} e^{-\frac{(k-2)^2}{2n}} \tag{24}$$

$$= 2 + \sqrt{\frac{n\pi}{2}}\, \text{erf}\left(\frac{r-2}{\sqrt{2n}}\right) = F(r) \tag{25}$$

Previous inequality steps follow easily from propositions 1, 2 and 3. Now we can express the precision in digits (d) with the equation

$$-\log_{10}\left(\frac{F(n) - F(r)}{\sqrt{\frac{n\pi}{2}}}\right) = d, \tag{26}$$

where the denominator inside the logarithm is a lower bound approximation for the binomial normalizing sum. For example Laplace's method gives for (6) the approximation (Flajolet and Sedgewick, 2005):

$$\mathcal{C}(2,n) = \sqrt{\frac{n\pi}{2}} + \frac{2}{3} + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right). \quad (27)$$

By omitting the constant term, we get our lower bound approximation for the denominator, which is valid for all data sizes (exact proof omitted). Thus we have

$$-\log_{10}\left(\frac{\sqrt{\frac{n\pi}{2}} - \sqrt{\frac{n\pi}{2}}\ \mathrm{erf}\left(\frac{r-2}{\sqrt{2n}}\right)}{\sqrt{\frac{n\pi}{2}}}\right) = d \quad (28)$$

$$-\log_{10}\left(1 - \ \mathrm{erf}\left(\frac{r-2}{\sqrt{2n}}\right)\right) = d. \quad (29)$$

We replaced the first error function in (26) with its supremum value 1 and got (28). If we solve r, we have

$$r = 2 + \sqrt{2n} \cdot R, \quad (30)$$

where

$$R = \mathrm{erf}^{-1}(1 - 10^{-d}). \quad (31)$$

The final task is now to approximate the inverse of the error function (Winitzky, 2008). We need this for approximating R to get a nice, clean and computable bound. First we compute the Taylor approximation:

$$g(x) = \ln(1 - \mathrm{erf}(x)^2) \approx -\frac{4}{\pi}x^2 + \mathcal{O}(x^4). \quad (32)$$

We need only the first non-zero term for our purpose. The approximation is then

$$\mathrm{erf}(x) = \sqrt{1 - e^{g(x)}} \approx \sqrt{1 - e^{-\frac{4x^2}{\pi}}}, \quad (33)$$

and it is good enough as our tests later show. This is not a lower bound approximation, but we need one, because we invert the approximating function. A little dirty trick solves our problem. We just change the multiplier 4 to π (Proposition 4). This new function can be easily inverted and the result therefore is

$$\mathrm{erf}^{-1}(u) = \sqrt{-\ln(1 - u^2)}. \quad (34)$$

The final step is to set $u = 1 - 10^{-d}$ and we have the result

$$R \approx \sqrt{-\ln(1 - (1 - 10^{-d})^2)} \quad (35)$$

$$= \sqrt{-\ln(2 \cdot 10^{-d} - 100^{-d})}. \quad (36)$$

□

We continue approximating R, because for the time complexity reasons, we need a simplified form to see the magnitude of this term. We easily see that

$$\sqrt{-\ln(2 \cdot 10^{-d} - 100^{-d})} \leq \sqrt{-\ln(10^{-d})} \quad (37)$$

$$= \sqrt{d\ln(10)}, \quad (38)$$

and therefore the required number of terms is $\mathcal{O}(\sqrt{dn})$.

The index bound seems to be quite good. In Figure 2 we have plotted optimal indexes and indexes given our bound with respect to data size n. We chose precisions so that they correspond approximately to single and double precision floating-point numbers. If n is one million, the index error is about $+250$ in both cases. The single precision error is a bit larger than the double precision error, because the index bound is getting tighter as precision increases.

4 The Multinomial Normalizing Sum

4.1 Properties of the Sum Terms

As we already saw, the ratio of the terms of the binomial normalizing sum is a simple rational function. In the multinomial case the ratio is the function

$$\frac{m_k}{m_{k-1}} = \frac{(n-k+1)(k+L-2)}{nk}. \quad (39)$$

Let us look at the terms of the multinomial normalizing sum. Figure 3 suggest that there is the biggest term and if we look at the term function, we see that it is unimodal. The next theorem and its proof give formal justifications for these claims.

Theorem 2. *The index of the biggest term of the multinomial normalizing sum is* $\left\lfloor \frac{1}{2}\left(3 - L + \sqrt{L^2 + (4n-2)L - 8n + 1}\right)\right\rfloor$.

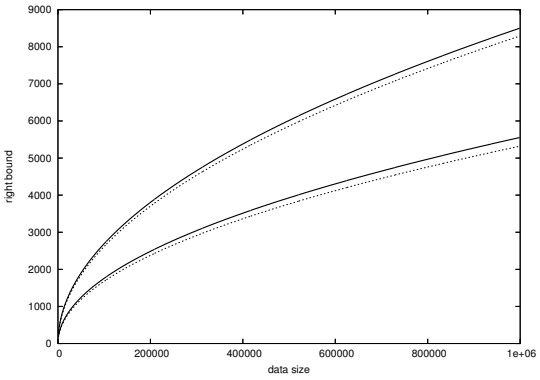


Figure 2: Terms needed for 16 (above) and 7 digit precisions with given data size. Actual approximations are shown as thick solid line. Thin dotted lines represent optimal index values.

Proof. We have to solve the equation

$$\frac{m_k}{m_{k-1}} = 1 \tag{40}$$

with respect to k. In other words, we are interested in values of real-valued k, where two consecutive sum terms have the same value. This requires solving roots of second order polynomial with respect to k. The other root is always negative and therefore is not in our range; let us denote the positive root by r. We are allowed to take the floor, because we know that the peak is between continuous index values $r-1$ and r. Outside this range all the sum term values are smaller than inside the range. Inside this range there are one or two integer points. If only one, then it is $\lfloor r \rfloor$. If there are two integer indexes, then $r-1$ and r must be these and they both give the same maximum value and thus $\lfloor r \rfloor$ gives the other of the two maximum values. □

This proved at the same time that the sum terms are getting bigger until they reach the peak, after which they start getting smaller. This unimodality gives us a great opportunity to construct simple and efficient algorithms.

4.2 About the Index Bounds

We can guess from the 15-nomial in Figure 3 that if we want to compute the sum in a fixed precision, we actually have to compute the sum terms from some $s \geq 0$ to some $t \leq n$. This means that the index of the first required term can be bigger than 0.

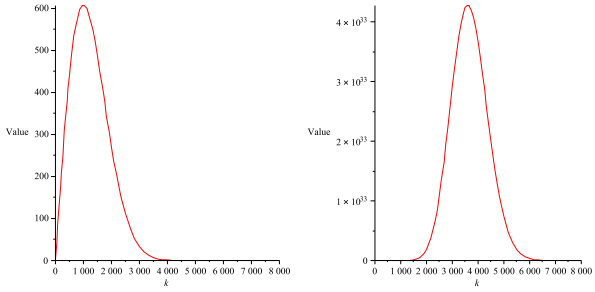


Figure 3: Magnitude of the first 8000 terms of the trinomial (left) and the 15-nomial (right) normalizing sums when data size (n) is one million.

However, we have to compute the factorials starting from 0 (to avoid any approximation), so we still have to start from index 0. Our empirical tests also show that we have to compute a lot more terms than in the binomial case. The effect is visible also by comparing figures 1 and 3. However, we can use (8) for computing the multinomial normalizing sum when the value of the binomial normalizing sum is known. This recurrence formula method seems to be much more efficient and therefore it is unnecessary to prove bounds in the multinomial case.

5 Sub-Linear Algorithms

First we present a simple algorithm (Algorithm 1), which is validated directly by unimodality. The basic idea is that the algorithm can sum the terms of a multinomial sum until the sum does not change anymore. Terms of the left slope always change the sum because each term is bigger than the previous one. After the peak, terms are monotonically getting smaller, and we know that the terms are never getting bigger anymore. Therefore the algorithm can stop after the sum has converged. All terms in the tail are so small and decaying so rapidly, that they cannot have a significant effect on the finite sum (we will return to this subject after the second algorithm). A precision of the result is determined by the precision of the used floating-point numbers.

Notice that the algorithm is using the recurrence

$$m_k = \frac{(n-k+1)(k+L-2)}{nk} \cdot m_{k-1} \tag{41}$$

while computing the sum terms. This way we can avoid huge factorial values and also floating-point

```
Compute_Multinomial(L,n){
  double sum = 1, previous_sum = -1, m = 1;
  int k = 1;

  for k from 1 to n by 1{
    m = (k+L-2) * (n-k+1) / (n*k) * m;
    sum = sum + m;

    if(sum is same as previous_sum){
      return sum;
    }

    previous_sum = sum;
  }

  return sum;
}
```

Algorithm 1: A simple algorithm for computing the multinomial normalizing sum.

```
Compute_Multinomial_With_Recurrence(L,n){
  double sum = 1, b = 1, old_sum, new_sum;
  int k,j;
  int bound = ceil( 2 + sqrt( -2*n*ln( 2*10^(-d)
                    -100^(-d) ) ) );

  for k from 1 to bound by 1{
    b = (n-k+1) / n * b;
    sum = sum + b;
  }

  old_sum = 1;

  for j from 3 to L by 1{
    new_sum = sum + (n * old_sum) / (j-2);
    old_sum = sum;
    sum = new_sum;
  }

  return sum;
}
```

Algorithm 2: A faster algorithm for computing the multinomial normalizing sum.

errors seem to be much lower. However the time complexity is not proved for this algorithm, as we did not compute the index bounds in the multinomial case. In addition, we cannot set the digit precision freely. For these reasons we present a second algorithm, which also seems to be twice as fast as the first one.

The intuitive idea of Algorithm 2 is to first compute the index bound, and then compute the binomial normalizing sum using the ratio of successive terms. After this, the algorithm uses the recurrence formula to compute the wanted multinomial normalizing sum. Variable `bound` is not directly computable as presented in the pseudocode, but we can use some standard logarithmic manipulation to avoid underflow. The time complexity of this second algorithm is $\mathcal{O}(\sqrt{dn} + L)$.

Now we can revise the question about the tail sum. Terms below the index bound do not affect the sum, but the first algorithm actually stops in the binomial case before the right bound is reached, because single terms do not change the sum. But if we take a sum starting from the stopping point all the way to the index bound, we can notice that this tail sum can affect the original sum. In our empirical tests we found out that the effect seems to be only on the few last digits, which is of the same magnitude as the floating point errors of our algorithms.

In Figure 4 we can see the average decay of the last digits with respect to data size. The variance is small between different numbers of same magnitude, so the figure gives a realistic impression. In the end we decided to keep the algorithms simple, because such small errors in the criterion hardly have any effect on the actual model selection task.

There are two operations that in fact increase precision. First we found out that the recurrence formula in Algorithm 2 tends to increase precision of those terms with an odd number of outcomes. Second we still have to take a logarithm of the normalizing sum, when computing actual stochastic complexity. The effect of the latter operation seem to be about one digit.

Precise values can be achieved using a simple trick: use a floating-point precision that is higher than the precision d, and crop the tail digits (e.g. quad precision for triple precision).

6 Conclusions

Stochastic complexity is an elegant, information-theoretic criterion for learning probabilistic graphical model structures. Although probabilistic in nature, it is fully objective and does not involve any hyperparameters which may be introduce problems in learning (Silander et al., 2007).

The fastest previously known algorithms for computing the multinomial stochastic complexity are $\mathcal{O}(n)$-algorithms. In this paper we showed that our previous hypergeometric representation of multinomial normalizing sums can be used for deriving sub-linear algorithms.

As the multinomial variable is an important building block in many more complex model classes, the results are directly applicable to model selec-

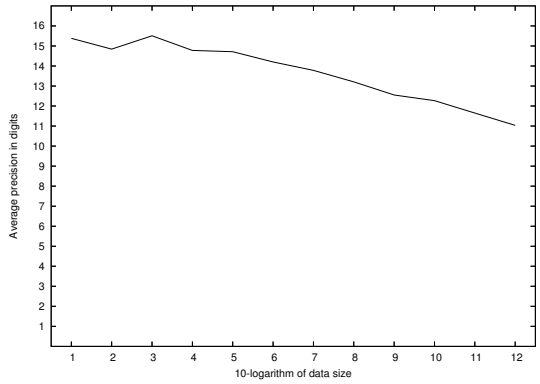


Figure 4: Average precision in the binomial normalizing sum with respect to data size using double precision floating-point numbers and Algorithm 2.

tion tasks in these cases as well. However, it should be noted that if the learning criterion is defined via the standard normalized maximum likelihood, the savings in the overall computational complexity are not necessarily very significant: for example, in the Naive Bayes case (classification or clustering tasks if the root node is hidden), the learning criterion involves a product of multinomial normalizing sums corresponding to the predictor variables, and these can be now computed in sub-linear time using the results above. Nevertheless, the real bottleneck of the computation process is a convolution operation, which still takes at least $\mathcal{O}(n^2 \log L)$ floating-point operations to compute (Mononen and Myllymäki, 2007). Similarly, when learning tree-structured Bayesian networks (Mononen and Myllymäki, 2008a), we can speed-up some parts of the computation, but not all.

On the other hand, there now exists a slightly different way to define the stochastic complexity, based on the factorized NML (fNML) criterion (Roos et al., 2008). The fNML criterion can be used in the Naive Bayes or in the Bayesian tree case, or even for learning Bayesian networks (DAGs) in general. In this case, the learning criterion factorizes into a product of multinomials, which means that the speed-up offered by our sub-linear algorithm is more apparent.

Acknowledgments

This work was supported in part by the Academy of Finland under the project Civi and by the Finnish Funding Agency for Technology and Innovation under the projects Kukot and PMMA. In addition, this work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence.

References

P. Flajolet and R. Sedgewick. 2005. *Analytic Combinatorics*. Unpublished.

P. Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press.

P. Kontkanen and P. Myllymäki. 2007. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6):227–233.

T. Mononen and P. Myllymäki. 2007. Fast NML computation for Naive Bayes models. In V. Corruble, M. Takeda, and E. Suzuki, editors, *Proceedings of the 10th International Conference on Discovery Science*, October.

T. Mononen and P. Myllymäki. 2008a. Computing the NML for Bayesian forests via matrices and generating polynomials. In *Proceedings of the IEEE Information Theory Workshop*, Porto, Portugal, May.

T. Mononen and P. Myllymäki. 2008b. On the multinomial stochastic complexity and its connection to the birthday problem. In *Proceedings of the International Conference on Information Theory and Statistical Learning*, Las Vegas, NV, July.

J. Rissanen. 2007. *Information and Complexity in Statistical Modeling*. Springer.

T. Roos, T. Silander, P. Kontkanen, and Myllymäki P. 2008. Bayesian network structure learning using factorized NML universal models. In *Proceedings of the Information Theory and Applications Workshop*, San Diego, CA, January.

Yu.M. Shtarkov. 1987. Universal sequential coding of single messages. *Problems of Information Transmission*, 23:3–17.

T. Silander, P. Kontkanen, and P. Myllymäki. 2007. On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. In R. Parr and L. van der Gaag, editors, *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 360–367. AUAI Press.

S. Winitzky. 2008. A handy approximation for the error function and its inverse. Unpublished.

Structural-EM for Learning PDG Models from Incomplete Data

Jens D. Nielsen
Computer Science Dept.
University of Castilla-La Mancha
Albacete, Spain
dalgaard@dsi.uclm.es

Rafael Rumí **Antonio Salmerón**
Statistic and Applied Mathematics Dept.
University of Almería
Almería, Spain
{rrumi|antonio.salmeron}@ual.es

Abstract

Probabilistic Decision Graphs (PDGs) are a class of graphical models that can naturally encode some context specific independencies that cannot always be efficiently captured by other popular models, such as Bayesian Networks. Furthermore, inference can be carried out efficiently over a PDG, in time linear in the size of the model. The problem of learning PDGs from data has been studied in the literature, but only for the case of complete data. In this paper we propose an algorithm for learning PDGs in the presence of missing data. The proposed method is based on the EM algorithm for estimating the structure of the model as well as the parameters. We test our proposal on artificially generated data with different rates of missing cells, showing a reasonable performance.

1 Introduction

The Probabilistic Decision Graph (PDG) model was first introduced by Bozga and Maler (1999), and was originally proposed as an efficient representation of probabilistic transition systems. In this study, we consider the more generalised version of PDGs proposed by Jaeger (2004).

PDGs constitute a class of probabilistic graphical models that can represent some context specific independencies that can not efficiently be captured by Bayesian network (BN) models. Also, probabilistic inference can be carried out directly in the PDG structure and has a time complexity linear in the size of the PDG model. This makes learning of PDGs especially interesting, as we are learning directly the inference structure, which is in contrast to the usual scenario when learning general BN models.

The performance of the PDG model w.r.t. general probability estimation has previously been studied and results suggest that the model in general performs competitively when compared to BN or Naïve BN models (Nielsen and Jaeger, 2006). The PDG model has also been successfully applied to supervised classification problems (Nielsen et al., 2007).

In this paper we are concerned with the estimation of PDGs from data. The problem has been addressed by Jaeger et al. (2006), where an algorithm based on the optimisation of a score is proposed for learning from complete data. However, the task of learning PDGs in the presence of missing data has not yet been explored in the literature. The difficulty arises in the computation of the score for a model given the database with missing values. A similar problem is found in the case of learning BNs from incomplete databases. Friedman (1997) addressed this problem by proposing an algorithm for estimating the structure of a BN model based on the Expectation-Maximisation (EM) principle (Dempster et al., 1977; Lauritzen, 1995).

We propose an algorithm for learning PDGs inspired by the proposal of Friedman (1997), based on the EM principle. Both the structure and the parameters are re-adjusted in each iteration of the algorithm. That is, the adjustments made to the structure are guided by the expected increase in some score metric, while the adjustments made to the parameters are guided by the expected likelihood of a completed version of the incomplete data.

2 Background and Notation

We will denote random variables by uppercase letters, e.g. X, and sets with boldface uppercase let-

ters, e.g. $\mathbf{X}$. When X_i is a discrete categorical random variable, we will by lowercase letter $x_{i,j}$ refer to the j'th state of X_i under some ordering. We will by $R(X_i)$ refer to the set of possible states of X_i, and by $R(\mathbf{X}) = \times_{X_i \in \mathbf{X}} R(X_i)$ when $\mathbf{X}$ is a set of variables. We will use r_i as a shorthand for $|R(X_i)|$. By lowercase bold letters we refer to joint states of sets of variables, e.g. $\mathbf{x} \in R(\mathbf{X})$. When $X_i \in \mathbf{X}$ and $\mathbf{x} \in R(\mathbf{X})$ we denote $\mathbf{x}[X_i]$ the projection of $\mathbf{x}$ onto coordinate X_i.

Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a directed graph structure with set of nodes $\mathbf{V} = \{V_1, \ldots, V_n\}$ and set of directed edges $\mathbf{E} \subset \mathbf{V} \times \mathbf{V}$. We will then by $ch_G(V_i)$ and $pa_G(V_i)$ refer the set of children of V_i and parents of V_i respectively in structure G, hence $ch_G(V_i) = \{V_j \in \mathbf{V} : (V_i, V_j) \in \mathbf{E}\}$ and $pa_G(V_i) = \{V_j \in \mathbf{V} : (V_j, V_i) \in \mathbf{E}\}$. A tree is a directed acyclic graph where one unique node $V_r \in \mathbf{V}$ is designated root and has no parents $pa_G(V_r) = \emptyset$ while all other nodes have exactly one parent. A forest structure is a set of such trees.

2.1 The Probabilistic Decision Graph Model

A PDG encodes a joint probability distribution over a set of categorical random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ by a factorisation defined by a structure over a set of local distributions.

Definition 2.1 (The PDG Structure)**.** *Let F be a forest structure over $\mathbf{X} = \{X_1, \ldots, X_n\}$. A* PDG-structure *$G = \langle \mathbf{V}, \mathbf{E} \rangle$ for $\mathbf{X}$ w.r.t. F is a set of* rooted *acyclic directed graphs over nodes $\mathbf{V}$, such that:*

1. *Each node $\nu \in \mathbf{V}$ represents a unique $X_i \in \mathbf{X}$ and all $X_i \in \mathbf{X}$ are represented by at least one node $\nu \in \mathbf{V}$. We will by $\nu_{i,j}$ refer to the j'th node representing X_i under some ordering of the set of nodes representing X_i.*

2. *For each node $\nu_{i,j}$, each possible state $x_{i,h}$ of X_i and each successor $X_k \in ch_F(X_i)$ there exists exactly* one *edge $(\nu_{i,j}, \nu_{k,l}) \in \mathbf{E}$ with label $x_{i,h}$, where $\nu_{k,l}$ is some node representing X_k.*

Let $X_k \in ch_F(X_i)$. By $succ(\nu_{i,j}, X_k, x_{i,h})$ we refer to the unique node $\nu_{k,l}$ representing X_k that is reached from $\nu_{i,j}$ by following the edge with label $x_{i,h}$.

Example 2.1. *A forest F over binary variables $\mathbf{X} = \{X_0, \ldots, X_7\}$ can be seen in Figure 1(a), and a PDG structure over $\mathbf{X}$ w.r.t. F in Figure 1(b). The labelling of nodes in the PDG-structure is indicated in subscripts and (redundant) by the dashed boxes, e.g., the nodes representing X_2 are $\{\nu_{2,0}, \nu_{2,1}\}$. Dashed edges correspond to edges labelled 0 and solid edges correspond to edges labelled 1, for instance $succ(\nu_{5,0}, X_6, 0) = \nu_{6,1}$.*

A PDG structure is instantiated by assigning to every node a local probability distribution over the variable that it represents. By a PDG model over discrete random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ we refer to a pair $\mathcal{G} = \langle G, \mathbf{\Theta} \rangle$ where G is a PDG structure over $\mathbf{X}$ and $\mathbf{\Theta}$ is an instantiation of G. We denote by $\mathbf{p}^{\nu_{i,j}}$ the local distribution assigned to node $\nu_{i,j}$, and by $p_{x_{i,h}}^{\nu_{i,j}}$ the probability for state $x_{i,h}$ in local distribution $\mathbf{p}^{\nu_{i,j}}$. The semantics of the local distribution $\mathbf{p}^{\nu_{i,j}}$ is defined by the path(s) leading to the node $\nu_{i,j}$ from the root, that is, how $\nu_{i,j}$ can be *reached*. Let G be a PDG structure over variables $\mathbf{X}$ w.r.t. forest F. A node $\nu_{i,j}$ in G is *reached* by $\mathbf{x} \in R(\mathbf{X})$ if

- $\nu_{i,j}$ is a root in G, or
- $X_i \in ch_F(X_k)$, $\nu_{k,l}$ is reached by $\mathbf{x}$ and $\nu_{i,j} = succ(\nu_{k,l}, X_i, \mathbf{x}[X_k])$.

By $reach_G(i, \mathbf{x})$ we denote the unique node representing X_i reached by $\mathbf{x}$ in PDG-structure G.

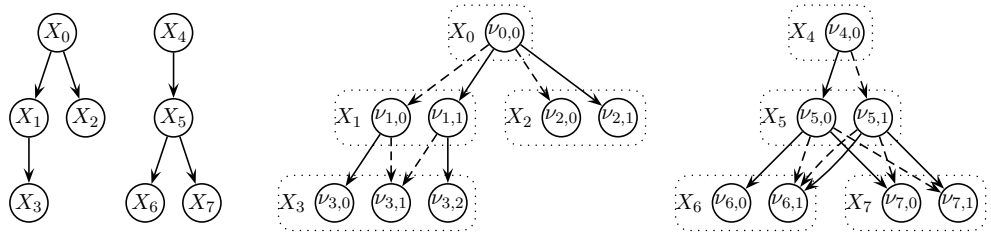
A PDG model $\mathcal{G} = \langle G, \mathbf{\Theta} \rangle$ over variables $\mathbf{X}$ represents a joint distribution $P^{\mathcal{G}}$ by the following factorisation:

$$P^{\mathcal{G}}(\mathbf{x}) = \prod_{X_i \in \mathbf{X}} p_{\mathbf{x}[X_i]}^{reach_G(i,\mathbf{x})}. \tag{1}$$

Example 2.2. *To instantiate the PDG structure in Fig. 1(b), we assign a local distribution to each node in the structure with the probabilistic interpretation given in Fig. 1(c). We can read some context specific independencies of this table, e.g. X_6 is independent of X_5 only in the context $X_4 = 0$.*

2.2 Selecting PDG models using complete data

For assessing models in the presence of observed data, we can use a penalised likelihood score function. Let $\mathcal{G}$ be a PDG model over variables $\mathbf{X} =$



(a) (b)

$\mathbf{p}^{\nu_{0,0}} = P(X_0)$	$\mathbf{p}^{\nu_{3,0}} = P(X_3 \mid X_0 = 0, X_1 = 1)$	$\mathbf{p}^{\nu_{5,1}} = P(X_5 \mid X_4 = 0)$
$\mathbf{p}^{\nu_{1,0}} = P(X_1 \mid X_0 = 0)$	$\mathbf{p}^{\nu_{3,1}} = P(X_3 \mid X_1 = 0)$	$\mathbf{p}^{\nu_{6,0}} = P(X_6 \mid X_4 = 1, X_5 = 1)$
$\mathbf{p}^{\nu_{1,1}} = P(X_1 \mid X_0 = 1)$	$\mathbf{p}^{\nu_{3,2}} = P(X_3 \mid X_0 = 1, X_1 = 1)$	$\mathbf{p}^{\nu_{6,1}} = P(X_6 \mid X_4 = 0 \vee \{X_4 = 1, X_5 = 0\})$
$\mathbf{p}^{\nu_{2,0}} = P(X_2 \mid X_0 = 0)$	$\mathbf{p}^{\nu_{4,0}} = P(X_4)$	$\mathbf{p}^{\nu_{7,0}} = P(X_7 \mid X_4 = X_5)$
$\mathbf{p}^{\nu_{2,1}} = P(X_2 \mid X_0 = 1)$	$\mathbf{p}^{\nu_{5,0}} = P(X_5 \mid X_4 = 1)$	$\mathbf{p}^{\nu_{7,0}} = P(X_7 \mid X_4 \neq X_5)$

(c)

Figure 1: A forest F over binary variables $\mathbf{X} = \{X_0, \ldots, X_7\}$ is shown in (a), and a PDG-structure over $\mathbf{X}$ w.r.t. variable forest F is shown in (b). In the PDG-structure in (b), solid edges are labelled with value 1 and dashed edges are labelled with value 0. In (c), we have indicated the probabilistic interpretation of the parameters for each node in the PDG structure of (b).

$\{X_1, \ldots, X_n\}$ and let $\mathbf{D}$ be a set of N complete observations of $\mathbf{X}$, then we define a general score function as:

$$S_\lambda(\mathbf{D}, \mathcal{G}) = (1 - \lambda) \cdot L(\mathbf{D}, \mathcal{G}) - \lambda \cdot size(\mathcal{G}) \,, \quad (2)$$

where $0 < \lambda < 1$, $size(\mathcal{G})$ is some measure of complexity of $\mathcal{G}$ and $L(\mathbf{D}, \mathcal{G})$ is the log-likelihood of $\mathbf{D}$ given $\mathcal{G}$. A typical definition of $size(G)$ is the number of free parameters in model $\mathcal{G}$. Using the following notation $\mathbf{D} = \{X_i^k : 1 \leq i \leq n, 1 \leq k \leq N\}$, the log-likelihood $L(\mathbf{D}, \mathcal{G})$ is:

$$\begin{aligned} L(\mathbf{D}, \mathcal{G}) &= \log \prod_{k=1}^{N} P^{\mathcal{G}}(X_1^k, \ldots, X_n^k) \\ &= \sum_{k=1}^{N} \log P^{\mathcal{G}}(X_1^k, \ldots, X_n^k) \\ &= \sum_{i=1}^{n} \sum_{h=1}^{r_i} \sum_{j=1}^{v_i} \#_{\mathbf{D}}(x_{i,h}, \nu_{i,j}) \log p_{x_{i,h}}^{\nu_{i,j}} \,, \quad (3) \end{aligned}$$

where v_i is the number of nodes representing X_i and $\#_{\mathbf{D}}(E)$ is the count of instances in $\mathbf{D}$ satisfying requirement E. For example, in Eq. (3) $\#_{\mathbf{D}}(x_{i,h}, \nu_{i,j})$ is the count of data items in $\mathbf{D}$ where variable X_i is observed in state $x_{i,h}$ and where the $\nu_{i,j}$ is reached.

3 Learning from Incomplete Data

Assume incomplete data, that is $\mathbf{D} = \mathbf{D}_O \cup \mathbf{D}_M$, where $\mathbf{D}_O$ is the elements of $\mathbf{D}$ containing a value and $\mathbf{D}_M = \mathbf{D} \setminus \mathbf{D}_O$. We can compute the expected log likelihood of $\mathbf{D}$ in model $\mathcal{G}$ (over variables $\mathbf{X}$), given some distribution P^* over $\mathbf{D}_M$ as:

$$\begin{aligned} &E[L(\mathbf{D}, \mathcal{G}) | \mathbf{D}_O, P^*] = \\ &\quad \sum_{i=1}^{n} \sum_{h=1}^{r_i} \sum_{j=1}^{v_i} E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,j})] \log p_{x_{i,h}}^{\nu_{i,j}} \,, \quad (4) \end{aligned}$$

where the second expectation also is with respect to $\mathbf{D}_O$ and P^*. In a structural EM algorithm like the one proposed by Friedman (1997), we optimise the expected likelihood instead of directly optimising the likelihood which in the presence of incomplete data no longer decomposes. Decomposability of the likelihood is important for model selection in which the search procedure in each step evaluates candidate models from a neighbourhood that is generated from a current model by local transformation. We define the expected score as a function Q:

$$\begin{aligned} &Q(\mathcal{G}, \mathbf{D} | \mathcal{G}^*) = \\ &\quad (1 - \lambda) E[L(\mathbf{D}, \mathcal{G}) | \mathbf{D}_O, \mathcal{G}^*] - \lambda size(\mathcal{G}) \,. \quad (5) \end{aligned}$$

In Eq. (5) we use the current model $\mathcal{G}^*$ as the reference distribution P^*. The structural EM procedure can now be stated as in Algorithm 1.

First, in line 5 of Alg. 1 we basically need to find MAP parameters for $\mathcal{G}$. Exact methods are usually intractable, so normally some approximation

Algorithm 1 The structural EM procedure

1: **procedure** SEM($\mathbf{D}$)
2: Let $\mathcal{G}^0 = \langle G^0, \mathbf{\Theta}^0 \rangle$ be the initial model.
3: $n \leftarrow 0$
4: **repeat**
5: $\mathbf{\Theta}^{n+1} \leftarrow \underset{Legal\ \mathbf{\Theta}}{argmax}\ Q(\langle G^n, \mathbf{\Theta} \rangle, \mathbf{D} | \mathbf{D}_O, \mathcal{G}^n)$
6: $G^{n+1} \leftarrow \underset{G \in \mathcal{N}(G^n)}{argmax}\ Q(\langle G, \cdot \rangle, \mathbf{D} | \mathbf{D}_O, \langle G^n, \mathbf{\Theta}^{n+1} \rangle)$
7: $\mathcal{G}^{n+1} \leftarrow \langle G^{n+1}, \mathbf{\Theta}^{n+1} \rangle$
8: $n \leftarrow n + 1$
9: **until** $Q(\mathcal{G}^n, \mathbf{D} | \mathcal{G}^{n-1}) \leq Q(\mathcal{G}^{n-1}, \mathbf{D} | \mathbf{D}_O, \mathcal{G}^{n-1})$
10: **return** $\mathcal{G}^{n-1}$

method is employed. Originally, Friedman (1997) proposed to use a standard EM approach in this step while Peña et al. (2000) propose as a more computationally efficient alternative to use the *branch and bound* procedure of Ramoni and Sebastiani (1997). However, the choice of approach in this step is not crucial to the following discussion in Sec. 4.

Second, in line 6 of Alg. 1, the function $\mathcal{N}(\cdot)$ is the neighbourhood generating function. We will define simple split and merge operations that implement structural modifications for generating neighbours from a current PDG model $\mathcal{G}$. We will show how to compute the expectations needed to evaluate the expected score of a neighbour.

4 SEM for PDG Models

In this section we will explain how Alg. 1 can be applied to PDG models. First, for constructing an initial model we need a forest structure over the variables in the domain. We accomplish this using the algorithm of Chow and Liu (1968) that induces a maximum weight spanning tree using mutual information as the edge weights. In Sec. 5 we explain how to compute mutual information from incomplete data.

Inducing the initial tree by finding a maximum weight spanning tree using mutual information as edge-weights, is different from previously proposed approaches for inducing variable forests/trees. In (Jaeger et al., 2006) a χ^2 test for conditional independence is used to assign marginally independent variables in different trees and conditionally independent variables in different sub-trees. In this study we use the above mentioned mutual information based method as it is less data-intensive compared to repeated χ^2 tests.

Assuming that we have the initial tree structure F over the variables, we initialise a PDG model as follows: for every variable $X_i \in \mathbf{X}$ with $pa_F(X_i) = X_k$, we create $v_i = r_k$ new nodes $\{\nu_{i,1}, \nu_{i,2}, \ldots, \nu_{i,v_i}\}$ representing X_i. We then connect every node $\nu_{k,j}$ representing X_k such that $succ(\nu_{k,j}, X_i, x_{k,z}) = \nu_{i,z}$. That is, for state $x_{k,z}$ of variable X_k the node $\nu_{i,z}$ is always reached. Constructing the initial PDG model in this way allows every variable to be modelled as marginally dependent on its parent and its set of children in F.

Finally, we use a random parametrisation $\mathbf{\Theta}^0$ of the initial structure G^0.

4.1 The Neighbourhood of a Model

In this subsection we explain how the neighbourhood $\mathcal{N}(G)$ of a PDG structure G is generated. We include operations that work on the PDG structure only, and leave the structure over the variables fixed. Operations that change the structure over the variables (e.g. operations that swap the position of two variables) are problematic as they potentially require the creation of a lot of new node connections. Offhand, it is not intuitive to us how to best do this in general, and therefore we choose to focus on the following two less dramatic structural changes that have both previously been used by Jaeger et al. (2006) for learning in the case of complete data[1].

Merging Nodes The merge operator takes a pair of nodes $\{\nu_{i,a}, \nu_{i,b}\}$ representing the same variable X_i. The nodes $\nu_{i,a}$ and $\nu_{i,b}$ are selected such that $succ(\nu_{i,a}, X_j, x_{i,h}) = succ(\nu_{i,b}, X_j, x_{i,h})$ for any state $x_{i,h} \in R(X_i)$ and child $X_j \in ch_F(X_i)$ in the variable forest F. The merge operation then simply consists in replacing nodes $\nu_{i,a}$ and $\nu_{i,b}$ with a new node $\nu_{i,c}$, where $\nu_{i,c}$ has as children exactly the children of $\nu_{i,a}$ (or $\nu_{i,b}$) and as parents inherits the union of parents of $\nu_{i,a}$ and parents of $\nu_{i,b}$.

Splitting Nodes The splitting operator takes as input a single node $\nu_{i,j}$ with m parents where $2 \leq m$, and replaces $\nu_{i,j}$ with m new nodes all representing X_i. Each new node inherits all the children of $\nu_{i,j}$,

[1] Jaeger et al. (2006) use an additional third operator that redirects edges. We leave this operator out of the algorithm for simplicity. Furthermore, for a given tree structure over the variables, any PDG structure can be reached from any other PDG structure using only merge and split operations.

and exactly one unique parent of $\nu_{i,j}$.

4.2 Scoring a Neighbour Model

In this section we detail how to compute the score $Q(\langle G', \cdot\rangle, \mathbf{D}|\mathbf{D}_O, \langle G, \mathbf{\Theta}\rangle)$ of a neighbour $G' \in \mathcal{N}(G)$ generated by merging two nodes or splitting a node. In fact, we will not compute the full expected score, but only the terms that are different.

4.2.1 Scoring a Merge Operation

Assume PDG $\mathcal{G}'$ is constructed from PDG $\mathcal{G} = \langle G, \Theta\rangle$ by merging nodes $\nu_{i,a}, \nu_{i,b} \in \mathbf{V}_i$ in structure G. Let the node $\nu_{i,c}$ be the one replacing $\nu_{i,a}$ and $\nu_{i,b}$ in $\mathcal{G}'$, and assume that we have computed (and stored) all expected counts of the form $E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,j})|\mathbf{D}_O, \mathcal{G}]$. Then, computing the expected counts for model $\mathcal{G}'$ under distribution $P^{\mathcal{G}}$ reduces to computing expected counts for $\nu_{i,c}$, which can be done efficiently from expectations $\#_{\mathbf{D}}(x_{i,h}, \nu_{i,a})$ and $\#_{\mathbf{D}}(x_{i,h}, \nu_{i,b})$, that is :

$$E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,c})|\mathbf{D}_O, \mathcal{G}] = E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,a})|\mathbf{D}_O, \mathcal{G}] + E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,b})|\mathbf{D}_O, \mathcal{G}]\,.$$

Hence, computing the difference in expected score $\Delta Q_{merge}(\nu_{i,a}, \nu_{i,b}) = Q(\mathcal{G}', \mathbf{D}|\mathbf{D}_O, \mathcal{G}) - Q(\mathcal{G}, \mathbf{D}|\mathbf{D}_O, \mathcal{G})$ reduces to computing the difference between the terms of the expected score involving nodes $\nu_{i,a}$ and $\nu_{i,b}$ and the new node $\nu_{i,c}$:

$$\begin{aligned}\Delta Q_{merge}(\nu_{i,a}, \nu_{i,b}) &= Q(\mathcal{G}', \mathbf{D}|\mathcal{G}) - Q(\mathcal{G}, \mathbf{D}|\mathcal{G}) \\ &= (1-\lambda)\left(\sum_{h=1}^{r_i} E[L_h^{\nu_{i,c}} - L_h^{\nu_{i,a}} - L_h^{\nu_{i,b}}|\mathbf{D}_O, \mathcal{G}]\right) \\ &\quad + \lambda\cdot(r_i - 1) \quad (6)\end{aligned}$$

where $\nu_{i,c}$ is the node resulting from merging $\nu_{i,a}$ and $\nu_{i,b}$, and $L_h^{\nu_{i,j}}$ is the term in the log-likelihood corresponding to node $\nu_{i,j}$ and the hth state of X_i. The expectation in (6) obviously can be computed term by term, and we see that $E[L_h^{\nu_{i,c}}|\mathbf{D}_O, \mathcal{G}] = E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,c})|\mathbf{D}_O, \mathcal{G}] \log E[p_{x_{i,h}}^{\nu_{i,c}}|\mathbf{D}_O, \mathcal{G}]$, where the expectation $E[p_{x_{i,h}}^{\nu_{i,c}}|\mathbf{D}_O, \mathcal{G}]$ is computed as the fraction $\frac{E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,c})|\mathbf{D}_O, \mathcal{G}]}{E[\#_{\mathbf{D}}(\nu_{i,c})|\mathbf{D}_O, \mathcal{G}]}$. The count $\#_{\mathbf{D}}(\nu_{i,a})$ is just $\sum_{h=1}^{r_i} \#_{\mathbf{D}}(x_{i,h}, \nu_{i,a})$.

The expectations $E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,j})|\mathbf{D}_O, \mathcal{G}]$ for any state $x_{i,h} \in R(X_i)$ and node $\nu_{i,j} \in \mathbf{V}_i$ are exactly the expectations we would compute in the parametric EM step in line 5 of Alg. 1. Therefore, these counts have already been computed for structure $\mathcal{G}^n$ in line 5 of Alg. 1, and can easily be made available at no extra cost.

4.2.2 Scoring a Split Operation

Let $\mathit{inc}(\nu_{i,j})$ be the set of edges incoming to $\nu_{i,j}$ in PDG structure $G = \langle \mathbf{V}, \mathbf{E}\rangle$, that is $\mathit{inc}(\nu_{i,j}) = \{(\nu_{k,z}, \nu_{i,j}) \in \mathbf{E}\}$. By $l_{\nu_{i,j}}^u$ we will denote the u'th element of $\mathit{inc}(\nu_{i,j})$ under some ordering. With $\nu_{i,j}^u$ we denote the node replacing $\nu_{i,j}$ for its uth incoming edge. Node $\nu_{i,j}$ is representing variable X_i and let the parent of X_i in the variable forest be X_k, hence by the definition of PDG structure, all parent nodes of $\nu_{i,j}$ represent variable X_k. The expected counts $E[\#_{\mathbf{D}}(\nu_{i,j}^u, x_{i,h})|\mathbf{D}_O, \mathcal{G}]$ for the node $\nu_{i,j}^u$ where $l_{\nu_{i,j}}^u = (\nu_{k,z}, \nu_{i,j})$ is labelled with state $x_{k,g}$ is then:

$$E[\#_{\mathbf{D}}(\nu_{i,j}^u, x_{i,h})|\mathbf{D}_O, \mathcal{G}] = E[\#_{\mathbf{D}}(\nu_{k,z}, x_{k,g}, x_{i,h})|\mathbf{D}_O, \mathcal{G}]\,. \quad (7)$$

The expectation in Eq. (7) can not be reconstructed from expected counts already computed for $\mathcal{G}$ in the structural parametric EM step of Alg. 1 (line 5) as was the case for the counts needed to evaluate a merge operation. However, anticipating that we will need such counts, we can store them during the computation of expectations in line 5 of Alg. 1. Assume that we have these expected counts available for structure G under the distribution defined by the PDG model $\mathcal{G} = \langle G, \Theta\rangle$. We can then compute the difference $\Delta Q_{\mathit{split}}(\nu_{i,j}) = Q(\mathcal{G}', \mathbf{D}|\mathcal{G}) - Q(\mathcal{G}, \mathbf{D}|\mathcal{G})$ for PDG model $\mathcal{G}'$ with structure G' generated by splitting node $\nu_{i,j}$ in structure G, as follows:

$$\begin{aligned}\Delta Q_{\mathit{split}}(\nu_{i,j}) &= Q(\mathcal{G}', \mathbf{D}|\mathcal{G}) - Q(\mathcal{G}, \mathbf{D}|\mathcal{G}) \\ &= (1-\lambda)\left[\sum_{h=1}^{r_i}\left(\sum_{u=1}^{m} E[L_h^{\nu_{i,j}^u}|\mathbf{D}_O, \mathcal{G}]\right) - E[L_h^{\nu_{i,j}}|\mathbf{D}_O, \mathcal{G}]\right] - \lambda(|\mathit{inc}(\nu_{i,j})| - 1)(r_i - 1)\,,\end{aligned} \quad (8)$$

where the log-likelihood terms $L_\cdot^\cdot$ are as described in Sec. 4.2.1. Further, it is clear that we can not split a root node as it is without parents.

4.3 Computing the Expectations

In order to compute the expected counts in sections 4.2.1 and 4.2.2, it is necessary to calculate probabilities of the form $P^{\mathcal{G}}(\{\nu \text{ is reached } \wedge X_i = x_i\}|\mathbf{Y} = \mathbf{y})$ for all $X_i \in \mathbf{X}$ and $\nu \in \mathbf{V}_i$, where $\mathcal{G}$ is a PDG over variables $\mathbf{X}$ and $\mathbf{y}$ is a joint observation of variables $\mathbf{Y} \subset \mathbf{X}$.

The computation of such probabilities can be done efficiently using the algorithm described by Jaeger (2004), which carries out the inference in time linear in the size of the PDG model. Broadly speaking, the desired probability is computed by first restricting the PDG $\mathcal{G}$ to $\mathbf{Y} = \mathbf{y}$. Then, for each node in the structure we compute parts of the product in (1) corresponding to incoming edges and outgoing edges. And, finally using these intermediate results stored in each node we can compute the desired conditional probabilities. We refer the reader to (Jaeger, 2004, Section 4) for details on PDG inference.

5 Estimating the Mutual Information with Missing Data

The mutual information between two random variables X and Y is defined as:

$$I(X,Y) = \sum_{i=1}^{|R(X)|} \sum_{j=1}^{|R(Y)|} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)} \quad . \tag{9}$$

As the joint distribution of X and Y is unknown, we need to estimate the mutual information from data. Assume we have a database $\mathbf{D}$ probably containing missing data. We require estimates for $\theta_{ij} = p(x_i, y_j)$, $\theta_{i.} = p(x_i)$ and $\theta_{.j} = p(y_j)$ for $i = 1, \ldots, |R(X)|$ and $j = 1, \ldots, |R(Y)|$. Actually, we only need to estimate θ_{ij}, since $\theta_{i.} = \sum_{j=1}^{|R(Y)|} \theta_{ij}$ and $\theta_{.j} = \sum_{i=1}^{|R(X)|} \theta_{ij}$.

Since $\mathbf{D}$ may contain missing data, we can use the EM algorithm to estimate the required parameters. The detailed procedure is given in Alg. 2.

Notice that steps 5 and 9 in algorithm 2 correspond, respectively, to the E and M steps of algorithm EM.

The value E_{ij} computed in line 7 of Alg. 2 is the expected number of records in $\mathbf{D}$ where X takes its i-th value and Y takes its j-th value. It is computed by exploring all the records $\mathbf{d} \in \mathbf{D}$ and calculating, for each record, the probability $P\{X = x_i, Y = y_j|\mathbf{d}, \mathbf{\Theta}^n\}$. That is, we compute:

$$E[\#_{\mathbf{D}}(X = x_i, Y = y_j)|\mathbf{\Theta}^n] = \sum_{\mathbf{d} \in \mathbf{D}} P\{X = x_i, Y = y_j|\mathbf{d}, \mathbf{\Theta}^n\} \quad . \tag{10}$$

The probability in Eq. (10) will be equal to 0 if the record has a value different to (x_i, y_j) and equal to 1 if the record is exactly equal to (x_i, y_j). If some of the cells in the record is missing, the probability is computed using the current estimates $\mathbf{\Theta}^n$.

Algorithm 2 EM for estimating the mutual information

1: **procedure** EM_MutualInformation($\mathbf{D}$)
2: Let $\mathbf{\Theta}^0 = \{\theta_{ij},\ i = 1, \ldots, |R(X)|,\ j = 1, \ldots, |R(Y)|\}$ be a random parametrisation of $p(x, y)$.
3: $n \leftarrow 0$.
4: **repeat**
5: **for all** $i = 1, \ldots, |R(X)|$ **do**
6: **for all** $j = 1, \ldots, |R(Y)|$ **do**
7: $E_{ij} \leftarrow E[\#_{\mathbf{D}}(X = x_i, Y = y_j)|\mathbf{\Theta}^n]$
8: $\mathbf{\Theta}^{n+1} \leftarrow \emptyset$
9: **for all** $i = 1, \ldots, |R(X)|$ **do**
10: **for all** $j = 1, \ldots, |R(Y)|$ **do**
11: $\theta_{ij}^{n+1} \leftarrow \frac{E_{ij}}{\sum_{k=1}^{|R(X)|} \sum_{l=1}^{|R(Y)|} E_{kl}}$
12: $\mathbf{\Theta}^{n+1} \leftarrow \mathbf{\Theta}^{n+1} \cup \{\theta_{ij}^{n+1}\}$
13: $n \leftarrow n + 1$.
14: **until** $L(\mathbf{D}|\mathbf{\Theta}^n) \leq L(\mathbf{D}|\mathbf{\Theta}^{n-1})$.
15: Estimate $I(X, Y)$ as:

$$\hat{I}(X,Y) = \sum_{i=1}^{|R(X)|} \sum_{j=1}^{|R(Y)|} \theta_{ij}^n \log \frac{\theta_{ij}^n}{\theta_{i.}^n \theta_{.j}^n} \quad .$$

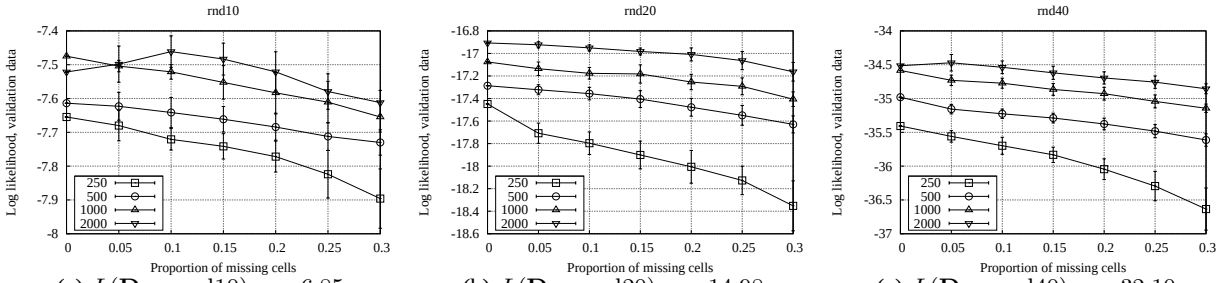
16: **return** $\hat{I}(X, Y)$.

6 Experiments

In order to test Alg. 1 we have performed experiments over different synthetic databases sampled from PDG models with 10, 20 and 40 variables generated at random[2]. We will refer to these models as rnd10, rnd20 and rnd40 respectively. From each PDG model, we constructed four databases containing 250, 500, 1000 and 2000 complete samples.

For each database, we have considered different rates of missing values, ranging from 5% to 30%. For each rate of missing values we generated

[2]The PDG models were generated at random with the restriction that they consist of a single variable tree.



(a) $L(\mathbf{D}_{\mathrm{val}}, \mathrm{rnd10}) = -6.85$ (b) $L(\mathbf{D}_{\mathrm{val}}, \mathrm{rnd20}) = -14.98$ (c) $L(\mathbf{D}_{\mathrm{val}}, \mathrm{rnd40}) = -32.10$

Figure 2: Mean and standard deviation of log-likelihood of a validation set of 10000 complete samples computed in the models leant from datasets sampled from model rnd10 (a), rnd20 (b) and rnd40 (c). The log-likelihood of the generating models are indicated beneath the plots.

50 databases from the original (complete) database by randomly erasing the value in a fraction of the cells according to the rate of missing values. The learning algorithm was then executed on each of the 50 databases measuring the quality of the learnt model as the log-likelihood of a separate validation database containing 10000 complete samples.

As score function we used the S_λ function of equation (2) with λ adjusted according to the size of the database to give a tradeoff between size and likelihood equivalent to the one imposed by the BIC score[3]. Finally, in order to speed up the algorithm, we put a limit of 10 iterations in each parametric EM[4] and 100 iterations in structural EM (the loop of Alg. 1).

6.1 Results

In Fig. 2(a-c) we show plots of mean and standard deviations of the log-likelihood of models learnt in the experiments described above.

First, the plots of Fig. 2 in general show the expected behaviour as mean likelihood generally decreases as a result of increasing the proportion of missing cells in the training data, while standard deviation increases. We note, also as expected, that the experiments on the larger data sets reach higher likelihood on the validation data and also show a more stable performance with less increase in standard deviation as the rate of missing values is increased.

[3]Setting $\lambda = \left(\frac{2N}{\log(N)} + 1\right)^{-1}$ where N is the number of observations yields BIC tradeoff.

[4]We run a 100 iterations parametric EM to optimise the parameters of the final model.

Second, in the experiment using 2000 samples from the rnd10 model (Fig. 2(a)) we observe an increase in likelihood up until a rate of 10% missing values. This behaviour may be caused by the algorithm over-fitting to the complete (rate 0% missing values) training data, while the introduction of some missing values helps the algorithm learn a less specific model with better ability to generalise. However, we only observe this behaviour for that specific data set which is somewhat unexpected assuming our explanation is valid.

Lastly, the initial tree structure is created using the classical algorithm of Chow and Liu (1968) as explained in Sect. 4. This initial model is itself a very commonly used model in probability estimation due to its simple restricted syntax and consequently efficient learning and inference. We therefore compare the quality of our final model to this initial model. From each experiment with missing data (72 total) we measured the likelihood of the validation data in the initial model as well as in the final PDG model. Using a Wilcoxon signed rank test for paired samples with significance level 0.05, we found significantly lower likelihood of the PDG model in 2 cases, no significant difference in 5 cases while in 65 cases we found significant better likelihood of the PDG model. The PDG performed significantly worse when using 500 samples of the rnd10 model with 25% and 30% missing cells, and no significant difference could be established for the experiments using the 250 samples of rnd10 with 30% missing, the 500 samples of rnd10 with 20% missing, the 250 samples of rnd20 with 30% missing, the 500 samples of rnd20 with 30% missing and

the 500 samples of rnd40 with 30% missing.

7 Concluding Remarks

In this paper we have proposed an algorithm for learning PDG models in the presence of missing data. Our proposal was inspired by previous work on learning BN models from incomplete data by Friedman (1997). We have tested our proposal on synthetic data sampled from randomly constructed PDG models. The experiments show that the algorithm performs well and behaves well even when the rate of missing cells are increased. Statistical tests shows significant improvement in quality over the initial Markov tree models in 65 out of the 72 experiments with incomplete data.

The algorithm introduced here can be extended in various ways. For instance, the use of other scores could be considered. Also, a Bayesian approach could be followed as in (Friedman, 1998).

We have only focused on the scenario where data is missing completely at random (MCAR). MCAR is the most general setting, and when data is truly MCAR, one could employ simpler and more efficient approaches to learning, such as available-case-analysis. We plan to investigate simpler and less general approaches in the future. Future studies also include the extension of the current algorithm to handle scenarios where unobserved variables are known to influence the observed data. Finally, a more exhaustive comparative analysis including other inference efficient graphical models (such as Naïve Bayes models) will be the focus of the next stage of this study.

Acknowledgements

This work has been supported by the Spanish Ministry of Education and Science, through projects TIN2007-67418-C03-01,02.

References

Bozga, M. and Maler, O. (1999). On the representation of probabilities over structured domains. In *Proceedings of the 11th International Conference on Computer Aided Verification*, pages 261–273. Springer.

Chow, C. K. and Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

Dempster, A. P., Laird, N. M., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Friedman, N. (1997). Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*.

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Proceedings of the UAI'98 Conference*.

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.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201.

Nielsen, J. D. and Jaeger, M. (2006). An empirical study of efficiency and accuracy of probabilistic graphical models. In *Proceedings of the Third European Workshop on Probabilistic Graphical Models*, pages 215–222.

Nielsen, J. D., Rumí, R., and Salmerón, A. (2007). El clasificador grafo de decisión probabilístico. Presented at: XXX Congreso Nacional de Estadística e Investigación Operativa. `http://www.ual.es/~dalgaard/publications/seio07.pdf`.

Peña, J. M., Lozano, J. A., and Larrañaga, P. (2000). An improved Bayesian structural EM algorithm for learning Bayesian networks for clustering. *Pattern Recognition Letters*, 21:779–786.

Ramoni, M. and Sebastiani, P. (1997). Learning Bayesian networks from incomplete databases. Technical Report KMI-TR-43, Knowledge Media Institute, The Open University.

Logical Properties of Stable Conditional Independence

Mathias Niepert and Dirk Van Gucht
Department of Computer Science
Indiana University
Bloomington, IN, USA

Abstract

We utilize recent results concerning a complete axiomatization of stable conditional independence (CI) relative to discrete probability measures to derive perfect model properties of stable CI structures. We show that stable CI can be interpreted as a generalization of undirected graphical models and establish a connection between sets of stable CI statements and propositional formulae in conjunctive normal form. Consequently, we derive that the implication problem for stable CI is coNP-complete. Finally, we show that SAT solvers can be employed to efficiently decide the implication problem and to compute non-redundant representations of stable CI, even for instances involving hundreds of variables.

1 Introduction

The importance of stable conditional independence for reducing the complexity of representation of conditional independence structures has recently been established (de Waal and van der Gaag, 2004). Stable CI is an alternative to graphical models in representing and reasoning with conditional independence. A good understanding of its logical and algorithmic properties could lead to new theoretical insights and applications in the field of uncertain reasoning and data mining. While several results regarding the characteristics of stable CI structures exist (Matúš, 1992)(de Waal and van der Gaag, 2004)(de Waal and van der Gaag, 2005), no study has investigated its logical properties as it was done for general CI and graphical models relative to the class of discrete probability measures (Geiger and Pearl, 1993). We use recent results concerning a complete axiomatization of stable CI relative to discrete probability measures (Niepert et al., 2008) to show that (1) stable CI has perfect models relative to discrete probability measures, (2) for some sets of stable CI statements there exists no perfect model relative to binary probability measures, and (3) the number of distinct stable CI structures grows at least double exponentially with the number of statistical variables. We also derive that stable CI structures can be interpreted as a generalization of undirected graphical models: for every UG model there exists a stable CI structure, and if a discrete probability measure is (perfectly) Markovian w.r.t. the UG model, then it satisfies (exactly) all the CI statements of the stable CI structure. We establish a direct connection between sets of stable CI statements and propositional formulae in conjunctive normal form and use this connection to show that the implication problem for stable conditional independence is coNP-complete. Finally, we show that existing SAT solvers can be employed to efficiently decide the implication problem and to compute non-redundant representations of stable CI, even for instances involving hundreds of variables.

2 Preliminaries

Definition 1. Throughout this paper, S will be a finite, implicit set of attributes (discrete statistical variables). The expression $I(A, B|C)$, with A, B, and C pairwise disjoint subsets of S, is called a *conditional independence (CI) statement*. If $ABC = S$, we say that $I(A, B|C)$ is *saturated*. If $A = \emptyset$ or $B = \emptyset$ or both, we say that $I(A, B|C)$ is *trivial*.

A1: $I(A, B|C) \rightarrow I(B, A|C)$
A2: $I(A, BD|C) \rightarrow I(A, D|C)$
A3: $I(A, B|CD) \wedge I(A, D|C) \rightarrow I(A, BD|C)$
A4: $I(A, B|C) \rightarrow I(A, B|CD)$
A5: $I(A, B|C) \wedge I(D, E|AC) \wedge I(D, E|BC) \rightarrow I(D, E|C)$

Figure 1: The inference rules of system $\mathcal{A}$.

The set of inference rules in Figure 1 will be denoted by $\mathcal{A}$. The *symmetry* (A1), *decomposition* (A2), and *contraction* (A3) rules are part of the semi-graphoid axioms (Pearl, 1988). *Strong union* (A4) and *strong contraction* (A5) are additional inference rules. The derivability of a CI statement c from a set of CI statements $\mathcal{C}$ under the inference rules of system $\mathcal{A}$ is denoted by $\mathcal{C} \vdash c$. The *closure* of $\mathcal{C}$ under $\mathcal{A}$, denoted $\mathcal{C}^+$, is the set $\{c \mid \mathcal{C} \vdash c\}$. Even though triviality is a sound inference rule, we will not mention it explicitly in the rest of the paper. Trivial CI statements are assumed to be implicitly present.

Definition 2. A *probability model* over $S = \{s_1, \ldots, s_n\}$ is a pair (dom, P), where dom is a domain mapping that maps each s_i to a finite domain $dom(s_i)$ and P is a probability measure having $dom(s_1) \times \cdots \times dom(s_n)$ as its sample space. For $A = \{a_1, \ldots, a_k\} \subseteq S$ we will say that $\mathbf{a}$ is a domain vector of A if $\mathbf{a} \in dom(a_1) \times \cdots \times dom(a_k)$. If $dom(s_i) = \{0, 1\}$ we say that the probability model is *binary*.

In what follows, we will only refer to probability measures, keeping their underlying probability models implicit. The class of discrete probability measures will be denoted by $\mathcal{P}$ and the class of binary probability measures by $\mathcal{B}$.

Definition 3. Let $I(A, B|C)$ be a CI statement and let P be a probability measure. We say that P *satisfies* $I(A, B|C)$ if for every domain vector $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ of A, B, and C, respectively, $P(\mathbf{c})P(\mathbf{a}, \mathbf{b}, \mathbf{c}) = P(\mathbf{a}, \mathbf{c})P(\mathbf{b}, \mathbf{c})$.

Relative to the notion of *satisfaction* we can now define the *conditional independence implication problem*.

Definition 4 (Probabilistic CI implication problem)**.** Let $\mathcal{C}$ be a set of CI statements, let c be a CI statement, and let $\mathcal{P}$ be the class of discrete probability measures. We say that $\mathcal{C}$ *implies* c relative to $\mathcal{P}$, and write $\mathcal{C} \models c$, if each measure $P \in \mathcal{P}$ that *satisfies* the CI statements in $\mathcal{C}$ also *satisfies* the CI statement c. The set $\{c \mid \mathcal{C} \models c\}$ will be denoted by $\mathcal{C}^*$.

A powerful tool in deriving results about the CI implication problem is the association of semi-lattices with CI statements (Niepert et al., 2008). Given subsets A and B of S we write $[A, B]$ for the lattice $\{U \mid A \subseteq U \subseteq B\}$.

Definition 5. Let $I(A, B|C)$ be a CI statement. The *semi-lattice* of $I(A, B|C)$ is defined by $\mathcal{L}(A, B|C) = [C, S] - ([A, S] \cup [B, S])$.

Example 1. Let $S = \{a, b, c, d\}$ and let $I(a, b|c)$ be a CI statement. The semi-lattice of this statement is $\{c, cd\}$.

We will often write $\mathcal{L}(c)$ to denote the semi-lattice of a CI statement c and $\mathcal{L}(\mathcal{C})$ to denote the union of semi-lattices, $\bigcup_{c' \in \mathcal{C}} \mathcal{L}(c')$, of a set of CI statements $\mathcal{C}$.

3 Stable Conditional Independence

When novel information is available to a probabilistic system, the set of associated, relevant CI statements changes dynamically. However, some of the CI statements will continue to hold. Stable CI can be thought of as a subclass of general CI: every set of stable CI statements is a set of CI statements. Some of the properties of stable CI were first investigated by Matúš (Matúš, 1992) who named it *ascending* conditional independence and later by de Waal and van der Gaag (de Waal and van der Gaag, 2004) who introduced the term *stable* conditional independence. Every set of CI statements can be partitioned into its *stable* and *unstable* part. In this section we recall an axiomatization of stable CI using inference rules and its relation to the *lattice-inclusion* property. We will use these results to show that stable CI has perfect models w.r.t. discrete probability measures, but not w.r.t. binary probability measures.

Definition 6. Let $\mathcal{C}$ be a set of CI statements, and let $\mathcal{C}^{SG+}$ be the semi-graphoid closure of $\mathcal{C}$. Then $I(A, B|C)$ is said to be *stable* in $\mathcal{C}$ if $I(A, B|C') \in \mathcal{C}^{SG+}$ for all sets C' with $C \subseteq C' \subseteq S$.

Definition 7. A stable CI structure is a set of stable conditional independence statements $\mathcal{C}$ such that $\mathcal{C} = \mathcal{C}^*$.

In the remainder of the paper, a set of stable CI statements will be *any* set of CI statements that are implicitly *known* to be stable. Hence, a set of stable CI statements $\mathcal{C}$ can be different from $\mathcal{C}^*$. We approach stable CI as a *structural representation* of conditional independence much like graphical models are possible representations of conditional independence. Now, let us turn to a crucial result for stable conditional independence. The inference system $\mathcal{A}$ was shown to be sound and complete for stable conditional independence (Niepert et al., 2008).

Theorem 1. *Let $\mathcal{C}$ be a set of stable CI statements and let c be a CI statement. Then the following statements are equivalent:*

(a) $\mathcal{C} \models c$*;*

(b) $\mathcal{C} \vdash c$*; and*

(c) $\mathcal{L}(\mathcal{C}) \supseteq \mathcal{L}(c)$*.*

Example 2. Let $S = \{a, b, d, e\}$, let $\mathcal{C} = \{I(a, b|\emptyset), I(d, e|a), I(d, e|b)\}$ be a set of stable CI statements, and let $c = I(d, e|\emptyset)$. We know by *strong contraction* that $\mathcal{C} \vdash c$ and, therefore, $\mathcal{C} \models c$ by Theorem 1. Now, $\mathcal{L}(\mathcal{C}) = \{\emptyset, d, e, de\} \cup \{a, ab\} \cup \{b, ab\} = \{\emptyset, a, b, d, e, ab, de\} \supseteq \{\emptyset, a, b, ab\} = \mathcal{L}(c)$.

Definition 8. Let $\mathcal{C}$ be a set of CI statements. A probability measure is a *perfect model* for $\mathcal{C}$ if it satisfies precisely the statements $\mathcal{C}^*$, that is, all the statements that are implied by $\mathcal{C}$ and none other.

The next result follows from the existence of discrete perfect models with respect to CI statements (Geiger and Pearl, 1993), a result which was later strengthened by (Peña et al., 2006).

Proposition 1. *For every set of stable CI statements $\mathcal{C}$ there exists a discrete probability measure P such that P satisfies exactly the statements in $\mathcal{C}^*$ and none other, that is, P is a perfect model for $\mathcal{C}$.*

The previous result does not hold for the class of binary probability measures and it follows that stable CI shares the perfect model properties with general CI.

Proposition 2. *There exists a set of stable CI statements $\mathcal{C}$ for which no binary probability model is perfect.*

Proof. Let $S = \{a, b, c\}$ and let $\mathcal{C} = \{I(a, b|\emptyset), I(a, b|c)\}$. Clearly, $\mathcal{C}$ is a set of stable CI statements. By Theorem 1(c) neither $I(a, c|\emptyset)$ nor $I(b, c|\emptyset)$ are implied by $\mathcal{C}$. From (Geiger and Pearl, 1993) we know that every binary probability measure that satisfies the elements in $\mathcal{C}$ also satisfies either $I(a, c|\emptyset)$ or $I(b, c|\emptyset)$. Thus, no binary probability measure is perfect for $\mathcal{C}$. □

4 Graphical Models and Stable CI

Our goal is to relate stable CI to graphical models and more specifically undirected graphical models. Ultimately, we will show that stable CI can be seen as a generalization of undirected graphical models. The following theorem establishes that the CI statements present in a Markov network form a stable CI structure.

Theorem 2. *Let G be a Markov network (i.e., an undirected graphical model) and let $\mathcal{C}(G)$ be the set of all CI statements* encoded *in G. Then $\mathcal{C}(G)$ is a stable CI structure.*

Proof. It is well-known that *strong union* is a sound inference rule for separation in undirected graphs (Pearl, 1988). In addition, it can be verified that the inference rule *strong contraction* is sound for undirected graph separation. Thus, inference system $\mathcal{A}$ is sound for separation in undirected graphs and the statement of the theorem follows. □

Corollary 1. *For every Markov network G there exists a stable CI structure $\mathcal{C}$ and every discrete probability measure that is (perfectly) Markovian w.r.t. G satisfies the elements in $\mathcal{C}$ (and none other).*

This shows that stable conditional independence can be interpreted as a generalization of Markov networks. In what follows, we investigate how much broader this representation is compared to graphical models in general. First,

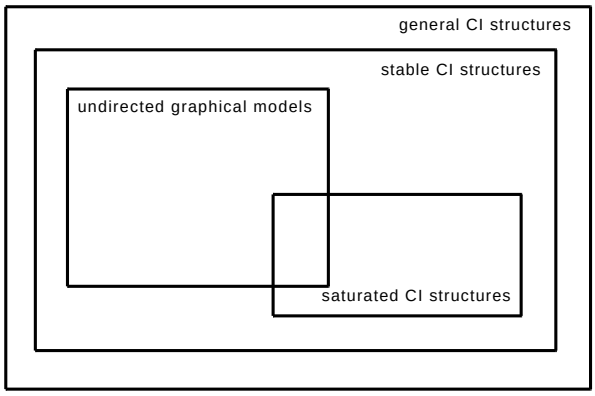


Figure 2: Inclusion relationships between different representations of conditional independence. Every undirected graphical model is a stable CI structure. Every saturated CI statements is trivially a stable CI statement.

we provide an example which demonstrates that there exists a stable CI structure that cannot be represented with a Markov network.

Example 3. Let $S = \{a, b, c, d\}$ and let $\mathcal{C} = \{I(a, b|cd), I(a, d|bc)\}$ be a set of stable CI statements. Note that by Theorem 1(c) no other CI statements are implied by $\mathcal{C}$ and hence, $\mathcal{C}$ is a stable CI structure. However, every Markov network that represents these two CI statements also represents the CI statement $I(a, bd|c)$ by the inference rule *intersection* which is sound for separation in undirected graphs (Pearl, 1988). Thus, the class of all CI structures induced by the class of Markov networks is a strict subclass of the class of stable CI structures.

Figure 2 depicts some relationships between different representations of conditional independence.

Proposition 3. *Let S be a finite set and let $x_i = \binom{|S|}{i}\binom{i}{2}$. The number of distinct stable CI structures over S is at least*

$$d_S =_{def} \sum_{i=2}^{|S|}(2^{x_i} - 1).$$

Proof. We sketch the proof. Let S be a finite set, let $V \subseteq S$ with $|V| = |S| - 2$, and let $U \subseteq V$. For every lattice $[U, V]$ there exists a stable CI structure $\mathcal{C}$ such that $\mathcal{L}(\mathcal{C}) = [U, V]$. Let $\ell = |S| - i$ for $2 \leq i \leq |S|$. Now, we have $2^{\binom{|S|}{i}\binom{i}{2}} - 1$ distinct combinations of lattices of the form $[U, V]$ with $|U| = \ell$ and each of these combinations represents a distinct stable CI structure by Theorem 1. □

Example 4. For $|S| = 3$ there are 8 UG, 22 discrete (Studený, 2005), and 14 stable discrete CI structures. For $|S| = 4$ there are 64 UG (Studený, 2005), 18478 discrete (Šimeček, 2006), and at least 4221 distinct stable CI structures. For $|S| = 5$ there are at least 2147485692 distinct stable CI structures, which is also a lower bound for the number of discrete CI structures.

As a consequence of Proposition 3 the number of stable CI structures grows double exponentially with the size of S.

5 Complexity of the Stable CI Implication Problem

In this section we will investigate the computational complexity of an important decision problems related to stable CI. Given a set of stable CI statements $\mathcal{C}$ and a CI statement c. Decide whether c is implied by $\mathcal{C}$. We will prove this decision problem to be `coNP`-complete. However, we will later show that a simple reduction to `UNSAT` exists. This allows one to make use of the many available `SAT` solvers and we will show experimentally that the problem can be decided very efficiently, even for instances involving hundreds of variables. We start with the formal definition of the decision problem.

Definition 9. Let $\mathcal{C}$ be a set of stable CI statements and let c be a CI statement. `STABLE-IMPLICATION` is the problem of deciding whether c is implied by $\mathcal{C}$, or, equivalently, whether the statement $\mathcal{C} \models c$ holds.

Lemma 1. `STABLE-IMPLICATION` *is in* `coNP`.

Proof. We show that the complement is in `NP`. Since $\mathcal{C} \not\models c$ if and only if $\mathcal{L}(\mathcal{C}) \not\supseteq \mathcal{L}(c)$ it is sufficient to find a $U \in \mathcal{L}(c)$ with $U \notin \mathcal{L}(\mathcal{C})$. This set can be guessed and then verified in polynomial time by checking for all $I(A, B|C) \in \mathcal{C}$ if $(U \not\supseteq C) \vee (U \supseteq A) \vee (U \supseteq B)$. □

We will now establish the correspondence between sets of stable CI statements and propositional formulae in conjunctive normal form, where a set of stable CI statement corresponds to a clause in the CNF formula and vice versa.

Definition 10. 3-CNFV is the set of all propositional formulae in conjunctive normal form with clauses of the form $x \vee y$, $\neg x \vee y \vee z$, $\neg x \vee \neg y \vee z$, and $\neg x \vee \neg y \vee \neg z$.

Proposition 4. *Let T be a set of propositional variables and let $\Phi \in$ 3-CNFV(T). Deciding whether Φ is satisfiable is* NP*-complete.*

Proof. This can be verified by a reduction from standard 3-CNF-SAT: the set of clauses we use in our construction are the clauses that occur in standard 3-CNF formulae except that every clause $x \vee y \vee z$ will be replaced by $(x \vee y \vee \neg w) \wedge (z \vee w)$, where w is a new variable. This reduction is possible in polynomial time and preserves satisfiability. □

Corollary 2. *Let T be a set of propositional variables and let $\Phi \in$ 3-CNFV(T). Deciding whether Φ is a contradiction is* coNP*-complete.*

Definition 11. Let T be a set of propositional variables and let X be a subset of T. The *minterm* associated with X, denoted $\mathbf{X}$, is the formula $\bigwedge_{a \in X} a \wedge \bigwedge_{b \in \overline{X}} \neg b$. Let Φ be a propositional formula over T. The *minset* of Φ, denoted $minset(\Phi)$, is the set $\{X \mid \mathbf{X} \models_{prop} \Phi\}$ where $\models_{prop}$ is the logical implication relation for propositional logic. The negative minset of Φ, denoted $negminset(\Phi)$, is the set $minset(\neg\Phi)$.

Definition 12. Let $T = \{t_1, ..., t_n\}$ be a set of propositional variables, let $\Phi \in$ 3-CNFV(T), let $\mathcal{C}(\Phi)$ be the set of clauses in Φ, let $S = T \cup \{r, s\}$ with $r \notin T$ and $s \notin T$, and let $\mathcal{T}(S)$ be the set of all non-trivial CI statements over S. Then f : 3-CNFV$(T) \rightarrow 2^{\mathcal{T}(S)}$ is defined as follows:

- $f(\Phi) = \bigcup_{c \in \mathcal{C}(\Phi)} f(c)$; with
- $f(t_i) = \{I(t_i, x|\emptyset) \mid x \in S - \{t_i\}\}$
- $f(\neg t_i) = \{I(x, y|t_i) \mid x, y \in S - \{t_i\}, x \neq y\}$
- $f(t_i \vee t_j) = \{I(t_i, t_j|\emptyset)\}$
- $f(\neg t_i \vee t_j) = \{I(t_j, x|t_i) \mid x \in S - \{t_i, t_j\}\}$
- $f(\neg t_i \vee \neg t_j) = \{I(x, y|\{t_i, t_j\}) \mid x, y \in S - \{t_i, t_j\}, x \neq y\}$
- $f(\neg t_i \vee t_j \vee t_k) = \{I(t_j, t_k|t_i)\}$
- $f(\neg t_i \vee \neg t_j \vee t_k) = \{I(t_k, x|\{t_i, t_j\}) \mid x \in S - \{t_i, t_j, t_k\}\}$
- $f(\neg t_i \vee \neg t_j \vee \neg t_k) = \{I(x, y|\{t_i, t_j, t_k\}) \mid x, y \in S - \{t_i, t_j, t_k\}, x \neq y\}$

Notice that the mapping f can be computed in polynomial time in the size of Φ and the number of variables involved. Furthermore, note that for any clause $c \in \mathcal{C}(\Phi)$ and for any $U \subseteq T$ we have $U \in \mathcal{L}(f(c))$ if and only if $\mathbf{U} \models_{prop} \neg c$.

Example 5. Let $T = \{a, b, c\}$, let $S = T \cup \{d, e\}$, and let $\Phi = (a \vee c) \wedge (\neg a \vee \neg b \vee c)$. Then $f(\Phi) = f(a \vee c) \cup f(\neg a \vee \neg b \vee c) = \{I(a, c|\emptyset)\} \cup \{I(c, d|ab), I(c, e|ab)\} = \{I(a, c|\emptyset), I(c, d|ab), I(c, e|ab)\}$ with $\mathcal{L}(f(\Phi)) = \{\emptyset, b, d, e, bd, be, bde, ab, abd, abe\}$ and $negminset(\Phi) = \{\emptyset, b, ab\}$.

Lemma 2. *Let T be a set of propositional variables, let $S = T \cup \{r, s\}$ with $r \notin T$, $s \notin T$, let f be the function from Definition 12, and let $\Phi \in$ 3-CNFV(T). Then we have the following:*

(1) $negminset(\Phi) \subseteq \mathcal{L}(f(\Phi))$; and

(2) Φ is a contradiction if and only if $\mathcal{L}(I(r, s|\emptyset)) \subseteq \mathcal{L}(f(\Phi))$.

Proof. To show (1) let $U \in negminset(\Phi)$. Then there exists a clause c in $\mathcal{C}(\Phi)$ such that $\mathbf{U} \models_{prop} \neg c$. But then for $I(x, y|U') \in f(c)$ it must be $U \supseteq U'$, $x \notin U$ and $y \notin U$ since otherwise $\mathbf{U} \models_{prop} c$. It follows that $U \in \mathcal{L}(f(c))$ and therefore $U \in \mathcal{L}(f(\Phi))$.

To show (2) let Φ be a contradiction. Notice that Φ is a contradiction if and only if $negminset(\Phi) = 2^T$. Now, $\mathcal{L}(I(r, s|\emptyset)) = 2^T = negminset(\Phi) \subseteq \mathcal{L}(f(\Phi))$, where the last inclusion follows from (1).

To show the other direction of (2) let $\mathcal{L}(I(r, s|\emptyset)) = 2^T \subseteq \mathcal{L}(f(\Phi))$. Assume that Φ is not a contradiction. Then there exists a set $U \subseteq T$ with $U \notin negminset(\Phi)$. Now, since $2^T \subseteq \mathcal{L}(f(\Phi))$ there must be a clause $c \in \mathcal{C}(\Phi)$ such that $U \in \mathcal{L}(f(c))$. Hence, $\mathbf{U} \models_{prop} \neg c$ and

Property	Stable CI
Complete finite axiomatization	Yes
Implication algorithm	coNP-complete
Perfect models [$\mathcal{P}$]	Yes
Perfect models [$\mathcal{B}$]	No

Figure 3: Summary of properties of stable CI.

thus $U \in negminset(\Phi)$, a contradiction to our assumption that $U \notin negminset(\Phi)$. □

Theorem 3. `STABLE-IMPLICATION` *is* `coNP`*-complete.*

Proof. Let T be a set of propositional variables, let $r \notin T$, $s \notin T$, and let $\Phi \in \texttt{3-CNFV}(T)$. Then, by Lemma 2 and Theorem 1, Φ is a contradiction if and only if $\mathcal{L}(f(\Phi)) \supseteq \mathcal{L}(I(r,s|\emptyset))$ if and only if $f(\Phi) \vdash I(r,s|\emptyset)$ if and only if $f(\Phi) \models I(r,s|\emptyset)$, where f is computable in polynomial time. Hence, `STABLE-IMPLICATION` is `coNP`-hard. The statement now follows from Lemma 1. □

The logical and algorithmic properties of stable CI are summarized in Figure 3.

6 Implication Testing and Redundancy Elimination Using SAT Solvers

In this section we will show that every set of stable CI statements can be reduced to a propositional formula. This allows us to employ SAT solvers to decide the implication problem and to compute irredundant equivalent subsets of stable CI structures. Stable CI can considerably reduce the size of representation of CI structures (de Waal and van der Gaag, 2004). First, we will define the notion of *irredundancy* and *redundancy* of representation for sets of stable CI statements. We will use terminology that was previously introduced in the context of propositional formulae in conjunctive normal form (Liberatore, 2005).

Definition 13. A set of stable CI statements $\mathcal{C}$ is *irredundant* if and only if $\mathcal{C} - \{c\} \not\models c$ for all $c \in \mathcal{C}$. Otherwise it is *redundant*.

A related definition is that of an irredundant equivalent subset. Note that a set of stable CI statements may have several different irredundant equivalent subsets and that the cardinality of these sets can differ.

Definition 14. Let $\mathcal{C}$ be a set of stable CI statements. A set of stable CI statements $\mathcal{C}'$ is an *irredundant equivalent subset* of $\mathcal{C}$ if and only if:

1. $\mathcal{C}' \subseteq \mathcal{C}$;
2. $\mathcal{C}' \models c$ for all $c \in \mathcal{C}$; and
3. $\mathcal{C}'$ is irredundant.

Example 6. Let $S = \{a, b, c\}$ and let $\mathcal{C} = \{I(a, b|\emptyset), I(a, b|c)\}$. Then, $\mathcal{C}' = \{I(a, b|\emptyset)\}$ is an irredundant equivalent subset of $\mathcal{C}$.

By Theorem 1 a stable CI structure can be derived from each of its irredundant equivalent subsets using the inference rules of system $\mathcal{A}$.

Definition 15. Let S be a finite set, let $\mathcal{C}$ be a set of CI statements, and let $I(A, B|C)$ be a CI statement. The mapping $g : 2^{\mathcal{I}(S)} \rightarrow \texttt{CNF}(S)$ is defined as

- $g(\mathcal{C}) = \bigwedge_{c \in \mathcal{C}}(g(c))$; with
- $g(I(A, B|C)) = \bigwedge_{a \in A} a \vee \bigwedge_{b \in B} b \vee \bigvee_{c \in C} \neg c$

The mapping g can be computed in linear time in the size of $\mathcal{C}$. Now, based on this mapping we can state the following theorem.

Theorem 4. *Let $\mathcal{C}$ be a set of stable CI statements and let c be a CI statement. Then the following statements are equivalent:*

- $\mathcal{C} \models c$*; and*
- $g(\mathcal{C}) \models_{prop} g(c)$.

Proof. We will again use the concepts *minset* and *negminset* introduced in Definition 11. Let $\mathcal{C}$ be a set of CI statements and let c be a CI statement. One can verify that $\mathcal{L}(\mathcal{C}) = negminset(g(\mathcal{C}))$ and $\mathcal{L}(c) = negminset(g(c))$.

irredundant-subset ($\mathcal{C}$: set) $\mathcal{C}'$: set

$\mathcal{C}' := \mathcal{C}$
for each $c \in \mathcal{C}'$
  begin
    if $g(\mathcal{C}' - \{c\}) \wedge \neg g(c)$ *not* satisfiable
      then $\mathcal{C}' := \mathcal{C}' - \{c\}$
  end
return $\mathcal{C}'$

Figure 4: A function to compute an irredundant equivalent subset.

By Theorem 1 we have that if $\mathcal{C}$ is a set of stable CI statements, then $\mathcal{C} \models c$ if and only if $\mathcal{L}(\mathcal{C}) \supseteq \mathcal{L}(c)$. Now, $\mathcal{L}(\mathcal{C}) \supseteq \mathcal{L}(c)$ if and only if $negminset(g(\mathcal{C})) \supseteq negminset(g(c))$ if and only if $g(\mathcal{C}) \models_{prop} g(c)$. □

Example 7. Let $S = \{a, b, d, e\}$, let $\mathcal{C} = \{I(a, b|\emptyset), I(d, e|a), I(d, e|b)\}$, and let $c = I(d, e|\emptyset)$. We have $g(\mathcal{C}) = (a \vee b) \wedge (d \vee e \vee \neg a) \wedge (d \vee e \vee \neg b)$ and $g(c) = d \vee e$. We also have $g(\mathcal{C}) \models_{prop} g(c)$ if and only if $g(\mathcal{C}) \wedge \neg g(c)$ is not satisfiable. Now, $g(\mathcal{C}) \wedge \neg g(c) = (a \vee b) \wedge (d \vee e \vee \neg a) \wedge (d \vee e \vee \neg b) \wedge \neg d \wedge \neg e$. This formula is not satisfiable. Hence, $\mathcal{C} \models c$ by Theorem 4.

Corollary 3. *Let $\mathcal{C}$ be a set of stable CI statements. Then $\mathcal{C}$ is irredundant if and only if for all c in $\mathcal{C}$ we have that $g(\mathcal{C} - \{c\}) \wedge \neg g(c)$ is satisfiable.*

The algorithm in Figure 4 is based on Corollary 3. It takes as input a set of stable CI statements $\mathcal{C}$ and returns an irredundant equivalent subset of $\mathcal{C}$ based on several satisfiability tests. For each number of attributes from 5 to 25 we randomly created sets of 500 CI statements and determined the size of the irredundant equivalent subsets using the algorithm. Figure 5 shows the average size of 1000 different runs. As one can expect, the fewer attributes there are the smaller is the irredundant equivalent subset.

The performance of the SAT solvers applied to instances of the implication problem was quite remarkable. We used MiniSat[1] by Niklas

[1] http://minisat.se

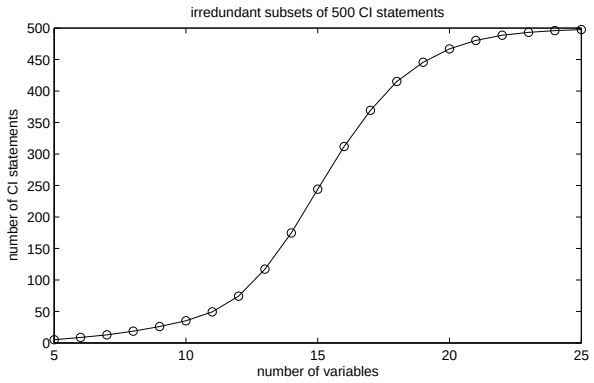


Figure 5: Size of irredundant equivalent subset of a set of initially 500 CI statements for different numbers of attributes.

variables	50	100	200	300	400
time [ms]	740	1523	3362	5627	7076

Figure 6: Average time needed (in milliseconds) to decide the implication problem for different numbers of variables and 100,000 antecedents.

Eén and Niklas Sörensson on a Pentium4 dual-core Linux system for the experiments. For the 500 satisfiability tests made to compute an irredundant equivalent subset, the algorithm took at most 1100 ms, where the majority of the time was spent on unsatisfiable instances of the problem. This amounts on average to 2ms per satisfiability test for sets of 500 CI statements.

In a second experiment we applied the SAT solver to larger, randomly generated instances of the stable CI implication problem with up to 400 variables. Figure 6 shows the average time (out of 10 tests) needed to decide the implication problem $\mathcal{C} \models c$ for $|\mathcal{C}| = 100,000$ and different numbers of variables.

7 Discussion and Future Work

We used a finite complete axiomatization of stable conditional independence to show that stable CI has the same perfect model properties as general conditional independence. In addition, we proved that stable conditional independence can be interpreted as a generalization or extension of undirected graphical models in that the

class of stable CI structures is a strict superset of the class of CI structures induced by undirected graphical models. Many procedures that learn graphical models are based on the *data faithfulness assumption*, see for example (Studený, 2005). The data faithfulness assumption states that data are "generated" by a probability measure P which is perfectly Markovian with respect to an instance of the class of graphical model under consideration. Now, learning methods based on these procedures are only safely applicable if the data faithfulness assumption is guaranteed.

While the data faithfulness assumption is also *not* guaranteed for the class of stable CI structures, we have as a consequence of Proposition 3 that the number of stable CI structures grows double exponentially with the size of S and, therefore, *more* probability measures are perfect with respect to a stable CI structure. On one hand, this implies that a reasonable graphical representation of stable CI is unlikely, using arguments similar to those made in (Studený, 2005) on page 63. On the other hand, it shows that the class of stable CI structures is the broadest and only double exponentially growing class of CI structures for which a complete finite axiomatization using inference rules and an implication algorithm are known. We also know that this class of CI structures includes the class of all CI structures induced by undirected graphical models and that there exists an interesting, direct connection to propositional logic. Furthermore, we have demonstrated that SAT solvers can be used to efficiently decide the implication problem for stable conditional independence, even for large numbers of variables. Future research should be concerned with the development of algorithms that can learn stable CI models from data and for *probabilistic* inference in the context of stable CI.

In addition to the aforementioned possible applications, stable CI can also be used as part of a probabilistic system to store information about conditional independencies more efficiently, using irredundant equivalent subsets computed by the algorithm in Figure 4.

Acknowledgments

We thank Marc Gyssens and Paul Purdom for helpful discussions and the anonymous reviewers for their valuable suggestions.

References

Peter R. de Waal and Linda C. van der Gaag. 2004. Stable independence and complexity of representation. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 112–119.

Peter R. de Waal and Linda C. van der Gaag. 2005. Stable independence in perfect maps. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 161–168.

Dan Geiger and Judea Pearl. 1993. Logical and algorithmic properties of conditional independence and graphical models. *The Annals of Statistics*, 21(4):2001–2021.

Paolo Liberatore. 2005. Redundancy in logic i: Cnf propositional formulae. *Artif. Intell.*, 163(2):203–232.

František Matúš. 1992. Ascending and descending conditional independence relations. In *Transactions of the 11th Prague Conference on Information Theory*, pages 189–200.

Mathias Niepert, Dirk Van Gucht, and Marc Gyssens. 2008. On the conditional independence implication problem: A lattice-theoretic approach. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 435–443.

Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc.

Jose Peña, Roland Nilsson, Johan Björkegren, and Jesper Tegnér. 2006. Reading dependencies from the minimal undirected independence map of a graphoid that satisfies weak transitivity. In *Proccedings of the European Workshop on Probabilistic Graphical Models*, pages 247–254.

Milan Studený. 2005. *Probabilistic Conditional Independence Structures*. Springer-Verlag.

Petr Šimeček. 2006. A short note on discrete representability of independence models. In *Proccedings of the European Workshop on Probabilistic Graphical Models*, pages 287–292.

A* Wars: The Fight for Improving A* Search for Troubleshooting with Dependent Actions

Thorsten J. Ottosen and Finn V. Jensen
Department of Computer Science
Aalborg University
9220 Aalborg, Denmark

Abstract

Decision theoretic troubleshooting combines Bayesian networks and cost estimates to obtain optimal or near optimal decisions in domains with inherent uncertainty. We use the well-known A* algorithm to find optimal solutions in troubleshooting models where different actions may fix the same fault. We prove that a heuristic function proposed by Vomlelová and Vomlel is monotone in models without questions, and we report on recent work on pruning. Furthermore, we experimentally investigate a hybrid approach where A* is combined with a method that sometimes avoid branching. The method is based on an analysis of the dependency between actions as suggested by Koca and Bilgiç.

1 Introduction

Imagine that you have a device which has been running well up to now, but suddenly it is malfunctioning. A set of *faults* $\mathcal{F}$ describes the possible causes to the problem. To fix the problem you have a set $\mathcal{A}$ of *actions*, which may fix the problem and a set $\mathcal{Q}$ of *questions*, which may help identifying the problem. Each action or question S has a positive *cost* $\mathrm{C_S}(\boldsymbol{\varepsilon})$ possibly depending on evidence $\boldsymbol{\varepsilon}$. Your task is to fix the problem as cheaply as possible. In this paper we do not consider questions.

When actions in a model can remedy sets of faults that overlap, we say that the model has *dependent actions*. Finding an optimal solution in models with dependent actions is of great practical importance since dependent actions can be expected to occur in many non-trivial domains. However, all non-trivial troubleshooting scenarios have been shown to be NP-hard—this includes models with dependent actions (Vomlelová, 2003).

Two different approaches have previously been used for finding optimal strategies: (Jensen et al., 2001) describes a branch & bound algorithm whereas (Vomlelová and Vomlel, 2003) describes an AO* algorithm. The AO* algorithm can be used for models with questions, but since a model without questions does not lead to AND-nodes in the search tree, we only need to consider the simpler A* algorithm (Hart et al., 1968) (Dechter and Pearl, 1985) for models with dependent actions.

We can summarize our troubleshooting assumptions as follows:

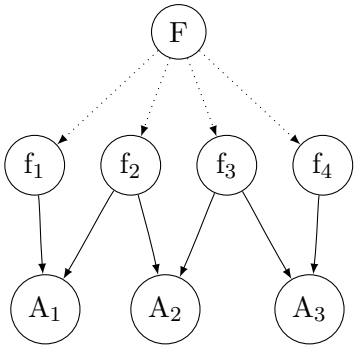
Assumption 1. *There are no questions.*

Assumption 2. *There is a single fault present when troubleshooting begins.* (This implies that we can have a single fault-node F with states $\mathrm{f}_i \in \mathcal{F}$.)

Assumption 3. *Actions are conditionally independent given evidence on the fault node.*

Assumption 4. *The cost of actions* $\mathrm{C_A}(\boldsymbol{\varepsilon})$ *is independent from evidence* $\boldsymbol{\varepsilon}$.

We use the following notation. The model provides for all $\mathrm{A} \in \mathcal{A}$ and $\mathrm{f} \in \mathcal{F}$ probabilities $\mathrm{P}(\mathrm{f}\,|\,\boldsymbol{\varepsilon})$, $\mathrm{P}(\mathrm{A}\,|\,\boldsymbol{\varepsilon})$ and $\mathrm{P}(\mathrm{A}\,|\,\mathrm{f})$, where $\boldsymbol{\varepsilon}$ is evidence. In Figure 1 is shown a simple model with dependent actions. We have some initial evidence $\boldsymbol{\varepsilon}^0$, and in the course of executing actions we collect further evidence. We write $\boldsymbol{\varepsilon}^i$ to denote that the first i actions have failed ($\boldsymbol{\varepsilon}^0 \subseteq \boldsymbol{\varepsilon}^i$), and we have by assumption $\mathrm{P}\big(\boldsymbol{\varepsilon}^0\big) = 1$ because



	f_1	f_2	f_3	f_4
$P(a_1 \mid F)$	1	1	0	0
$P(a_2 \mid F)$	0	1	1	0
$P(a_3 \mid F)$	0	0	1	1
$P(F)$	0.20	0.25	0.40	0.15
$C_{A_1} = C_{A_2} = C_{A_3} = 1$				

Figure 1: Left: a simple model for a troubleshooting scenario with dependent actions. The dotted lines indicate that the faults f_1 to f_4 are states in a single fault node F. A_1, A_2 and A_3 represent actions, and parents of an action node A are faults which may be fixed by A. Right: the quantitative part of the model.

the device is faulty. When action A has failed, we write $A = \neg a$ whereas $A = a$ means that it has succeeded. We often abbreviate $P(A = \neg a)$ as $P(\neg a)$. The presence of the fault f is written $F = f$, but we often abbreviate the event simply as f. Furthermore, we write $P(\boldsymbol{\varepsilon} \cup f)$ when we really had to write $P(\boldsymbol{\varepsilon} \cup \{f\})$. The set of faults that can be repaired by an action A is denoted $\mathrm{fa}(A)$. For example, in Figure 1 we have $\mathrm{fa}(A_2) = \{f_2, f_3\}$. In models where actions can have $P(a \mid \boldsymbol{\varepsilon}) = 1$, $\mathrm{fa}(\cdot)$ is a dynamic entity which we indicate by writing $\mathrm{fa}(\cdot \mid \boldsymbol{\varepsilon})$. The set of remaining actions is denoted $\mathcal{A}(\boldsymbol{\varepsilon})$, and $\mathcal{A}(f \mid \boldsymbol{\varepsilon}) \subseteq \mathcal{A}(\boldsymbol{\varepsilon})$ is the set of remaining actions that can fix f.

When there are no questions, a *troubleshooting strategy* is a sequence of actions $s = \langle A_1, \ldots, A_n \rangle$ prescribing the process of repeatedly performing the next action until the problem is fixed or the last action has been performed. To compare sequences we use the following definition:

Definition 1. The *expected cost of repair* (ECR) of a troubleshooting sequence $s = \langle A_1, \ldots, A_n \rangle$ with costs C_{A_i} is the mean of the costs until an action succeeds or all actions have been performed:

$$\mathrm{ECR}(s) \;=\; \sum_{i=1}^{n} C_{A_i}(\boldsymbol{\varepsilon}^{i-1}) \cdot P\big(\boldsymbol{\varepsilon}^{i-1}\big) \;.$$

We then define an *optimal sequence* as a sequence with minimal ECR. Also, $\mathrm{ECR}^*(\boldsymbol{\varepsilon})$ is the ECR for an optimal sequence of the actions $\mathcal{A}(\boldsymbol{\varepsilon})$ given $\boldsymbol{\varepsilon}$.

Example 1. Consider a sequence for the model in Figure 1:

$$\begin{aligned}
&\mathrm{ECR}(\langle A_2, A_3, A_1 \rangle) = \\
&\qquad C_{A_2} + P(\neg a_2) \cdot C_{A_3} + P(\neg a_2, \neg a_3) \cdot C_{A_1} \\
&= \; C_{A_2} + P(\neg a_2) \cdot C_{A_3} \\
&\qquad + P(\neg a_2) \cdot P(\neg a_3 \mid \neg a_2) \cdot C_{A_1} \\
&= \; 1 + \frac{7}{20} \cdot 1 + \frac{7}{20} \cdot \frac{4}{7} \cdot 1 = 1.55 \;.
\end{aligned}$$

A crucial definition is that of efficiency:

Definition 2. The *efficiency* of an action A given evidence $\boldsymbol{\varepsilon}$ is

$$\mathrm{ef}(A \mid \boldsymbol{\varepsilon}) \;=\; \frac{P(A = a)}{C_A(\boldsymbol{\varepsilon})} \;.$$

2 Monotonicity of the heuristic function ECR

A* (and AO*) is a best-first search algorithm that works by continuously expanding a frontier node n for which the value of the *evaluation function*

$$f(n) \;=\; g(n) + h(n), \qquad (1)$$

is minimal until finally a goal node t is expanded. The cost between two nodes n and m (m reachable from n) is denoted $c(n, m)$, and the function $g(n)$ is the cost from the start node s to n whereas $h(n)$ is the *heuristic function* that guides (or misguides) the search by estimating the cost from n to a goal node t.

If $h(n) \equiv 0$, A* degenerates to Dijkstra's algorithm. The cost of the shortest path from s to n is denoted $g^*(n)$, and from n to t it is denoted $h^*(n)$. Finally, the evidence gathered about actions from s to n is denoted $\boldsymbol{\varepsilon}^n$ ($\boldsymbol{\varepsilon}^0 \subseteq \boldsymbol{\varepsilon}^n$).

Definition 3. A heuristic function $h(\cdot)$ is *admissible* if

$$h(n) \quad \leq \quad h^*(n) \quad \forall n \ .$$

When A* is guided by an admissible heuristic function, it is guaranteed to find an optimal solution (Hart et al., 1968).

(Vomlelová and Vomlel, 2003) have suggested the following heuristic function for use in troubleshooting:

Definition 4. Let $\mathcal{E}$ denote the set containing all possible evidence. The function $\underline{\mathrm{ECR}} : \mathcal{E} \mapsto \mathcal{R}^+$ is defined for each $\boldsymbol{\varepsilon}^n \in \mathcal{E}$ as

$$\underline{\mathrm{ECR}}(\boldsymbol{\varepsilon}^n) = \mathrm{P}(\boldsymbol{\varepsilon}^n) \cdot \sum_{\mathrm{f} \in \mathcal{F}} \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f}) \ .$$

Remark. In (Vomlelová and Vomlel, 2003) the factor $\mathrm{P}(\boldsymbol{\varepsilon}^n)$ is left out. However, the factor ensures that the decomposition in Equation 1 takes the simple form

$$f(n) \quad = \quad \underbrace{\mathrm{ECR}\,(\boldsymbol{\varepsilon}^{\mathrm{n}})}_{\mathrm{g(n)}} + \underbrace{\underline{\mathrm{ECR}}(\boldsymbol{\varepsilon}^n)}_{\mathrm{h(n)}} \ ,$$

where $\mathrm{ECR}\,(\boldsymbol{\varepsilon}^{\mathrm{n}})$ is the ECR for the sequence defined by the path up to n. We also define

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}) \quad = \quad \sum_{\mathrm{f} \in \mathcal{F}} \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon} \cup \mathrm{f}) \ .$$

The optimal cost $\mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f})$ is easy to calculate under Assumption 2-4: the optimal sequence is found by ordering the actions in $\mathcal{A}\,(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^{\mathrm{n}})$ with respect to descending efficiency (Kadane and Simon, 1977).

Example 2. Assume the fault f can be repaired by two actions A_1 and A_2 and that $\mathrm{P}(\mathrm{a}_1 \,|\, \mathrm{f}) = 0.9$ and $\mathrm{P}(\mathrm{a}_2 \,|\, \mathrm{f}) = 0.8$. Furthermore, let both actions have cost 1. Since instantiating the fault node renders the actions conditionally independent, $\mathrm{P}(\mathrm{a} \,|\, \boldsymbol{\varepsilon} \cup \mathrm{f}) = \mathrm{P}(\mathrm{a} \,|\, \mathrm{f})$ and the efficiencies of the two actions are 0.9 and 0.8, respectively. We get

$$\begin{aligned} \mathrm{ECR}^*(\boldsymbol{\varepsilon} \cup \mathrm{f}) &= \mathrm{ECR}\,(\langle \mathrm{A}_1, \mathrm{A}_2 \rangle) \\ &= \mathrm{C}_{\mathrm{A}_1} + \mathrm{P}(\neg \mathrm{a}_1 \,|\, \mathrm{f}) \cdot \mathrm{C}_{\mathrm{A}_2} \\ &= 1 + 0.1 \cdot 1 = 1.1 \ . \end{aligned}$$

Not only is $\underline{\mathrm{ECR}}(\cdot)$ easy to compute, it also has the following property (Vomlelová and Vomlel, 2003):

Theorem 1. *Under Assumption 2-4 the heuristic function* $\underline{\mathrm{ECR}}(\cdot)$ *is admissible, that is,*

$$\underline{\mathrm{ECR}}(\boldsymbol{\varepsilon}^n) \quad \leq \quad \mathrm{ECR}^*(\boldsymbol{\varepsilon}^n) \quad \forall \boldsymbol{\varepsilon}^n \in \mathcal{E}.$$

For a class of heuristic functions, A* is guaranteed to have found the optimal path to a node when the node is expanded (Hart et al., 1968):

Definition 5. A heuristic function $h(\cdot)$ is *monotone* if

$$h(n) \quad \leq \quad c(n, m) + h(m),$$

whenever m is a successor node of n.

Remark. Monotonicity is equivalent to the often used and seemingly stronger *consistency* property: $h(n) \leq c^*(n, m) + h(m)\ \forall n, m$.

Henceforth we let A_n denote the performed action on the edge from a node n to a successor node m in the search graph.

Proposition 1. *For the heuristic function* $\underline{\mathrm{ECR}}(\cdot)$ *under Assumption 1 and 4 monotonicity is equivalent to*

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) \quad \leq \quad \mathrm{C}_{\mathrm{A}_n} + \mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m) \, .$$

Proof. We have $c(n, m) = \mathrm{P}(\boldsymbol{\varepsilon}^n) \cdot \mathrm{C}_{\mathrm{A}_n}$ and $\mathrm{P}(\boldsymbol{\varepsilon}^m) = \mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \mathrm{P}(\boldsymbol{\varepsilon}^n)$ and so the common factor $\mathrm{P}(\boldsymbol{\varepsilon}^n)$ cancels out. □

Theorem 2. *Under Assumption 1-4 the heuristic function* $\underline{\mathrm{ECR}}(\cdot)$ *is monotone.*

Proof. The idea is to express $\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m)$ in terms of $\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n)$. To do that we consider the complement of the set $\mathrm{fa}(\mathrm{A}_n)$ which is the set of all faults that A_n cannot fix. For each $\mathrm{f} \in \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)$ Bayes' rule (conditioned) yields

$$\mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^m) \quad = \quad \frac{1 \cdot \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n)}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)},$$

because $\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n \cup \mathrm{f}) \equiv 1$. If we restrict $\underline{\mathrm{ECR}}_h(\cdot)$ to a subset of faults X, we shall abuse notation and write it

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon} \,|\, X) = \sum_{\mathrm{f} \in X} \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon} \cup \mathrm{f}) \ .$$

In particular, we must have

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) = \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)) + \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathrm{fa}(\mathrm{A}_n)). \quad (2)$$

We can furthermore define

$$\begin{aligned} \Delta\mathcal{F} &= \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m \,|\, \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)) \\ &\quad - \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)) \ , \end{aligned}$$

which is an extra cost because all faults in $\mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)$ are more likely. Similarly

$$\Delta\mathrm{fa}(\mathrm{A}_n) = \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m \,|\, \mathrm{fa}(\mathrm{A}_n)) - \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathrm{fa}(\mathrm{A}_n)) \ ,$$

is the cost lost or gained because A_n has been performed and can no longer repair the faults $\mathrm{fa}(\mathrm{A}_n)$. We can then express $\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m)$ by

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m) = \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) + \Delta\mathrm{fa}(\mathrm{A}_n) + \Delta\mathcal{F}, \quad (3)$$

The constant $\mathrm{ECR}^*(\cdot)$ factors implies

$$\Delta\mathcal{F} = \sum_{\mathrm{f} \in \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)} [\mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^m) - \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n)] \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f}).$$

Exploiting Bayes' rule (as explained above) and Equation 2 we get

$$\Delta\mathcal{F} = \left[\frac{1}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)} - 1\right] \cdot \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)) = \left[\frac{1}{\mathrm{P}(\neg \mathrm{a_n} \,|\, \boldsymbol{\varepsilon}^n)} - 1\right] \cdot \left[\underline{\mathrm{ECR}}_\mathrm{h}(\boldsymbol{\varepsilon}^\mathrm{n}) - \underline{\mathrm{ECR}}_\mathrm{h}(\boldsymbol{\varepsilon}^\mathrm{n} \,|\, \mathrm{fa}(\mathrm{A_n}))\right].$$

Inserting into Equation 3 yields

$$\begin{aligned} \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m) &= \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) + \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m \,|\, \mathrm{fa}(\mathrm{A}_n)) - \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathrm{fa}(\mathrm{A}_n)) \\ &\quad + \left[\frac{1}{\mathrm{P}(\neg \mathrm{a_n} \,|\, \boldsymbol{\varepsilon}^\mathrm{n})} - 1\right] \cdot \left[\underline{\mathrm{ECR}}_\mathrm{h}(\boldsymbol{\varepsilon}^\mathrm{n}) - \underline{\mathrm{ECR}}_\mathrm{h}(\boldsymbol{\varepsilon}^\mathrm{n} \,|\, \mathrm{fa}(\mathrm{A_n}))\right] \\ &= \frac{\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n)}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)} + \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m \,|\, \mathrm{fa}(\mathrm{A}_n)) - \frac{1}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)} \cdot \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathrm{fa}(\mathrm{A}_n)), \end{aligned}$$

and we rearrange the equation into

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) = \mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \underline{\mathrm{ECR}}(\boldsymbol{\varepsilon}^m) + \underbrace{\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n | \mathrm{fa}(\mathrm{A}_n)) - \mathrm{P}(\neg \mathrm{a}_n | \boldsymbol{\varepsilon}^n) \cdot \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m | \mathrm{fa}(\mathrm{A}_n))}_{\Delta}.$$

By Proposition 1, we have to prove $\Delta \leq \mathrm{C}_{\mathrm{A}_n}$. Because of Bayes' rule and Assumption 3 we have

$$\begin{aligned} &\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^m) = \\ &\qquad \mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \frac{\mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n)}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)} \\ &\quad = \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n) \ . \end{aligned}$$

So we get

$$\Delta = \sum_{\mathrm{f} \in \mathrm{fa}(\mathrm{A}_n)} \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n) \cdot \underbrace{[\mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f}) - \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon}^m \cup \mathrm{f})]}_{\delta}.$$

Because of the single-fault assumption, we only need to prove that $\delta \leq \mathrm{C}_{\mathrm{A}_n}$. We now index the actions in $\mathcal{A}\,(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^\mathrm{n})$ as follows:

$$\frac{\mathrm{P}(\mathrm{B}_i = \mathrm{b}_i \,|\, \mathrm{f})}{\mathrm{C}_{\mathrm{B}_i}} \geq \frac{\mathrm{P}(\mathrm{B}_{i+1} = \mathrm{b}_{i+1} \,|\, \mathrm{f})}{\mathrm{C}_{\mathrm{B}_{i+1}}} \quad \forall i.$$

In this ordering, we have $\mathrm{A}_n = \mathrm{B}_x$. The inequalities generalizes to

$$\mathrm{C}_{\mathrm{B}_i} \leq \frac{\mathrm{P}(\mathrm{B}_i = \mathrm{b}_i \,|\, \mathrm{f})}{\mathrm{P}(\mathrm{B}_j = \mathrm{b}_j \,|\, \mathrm{f})} \cdot \mathrm{C}_{\mathrm{B}_j} \quad \forall j > i \ . \quad (4)$$

In particular, this is true for $j = x$ which we shall exploit later.

Assume we have N dependent actions in $\mathcal{A}\,(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^\mathrm{n})$. The first term of δ is then

$$\begin{aligned} \mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f}) &= \mathrm{ECR}^*(\langle \mathrm{B}_1, \ldots, \mathrm{B}_N \rangle) \\ &= \mathrm{C}_{\mathrm{B}_1} + \sum_{i=2}^{N} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \ . \end{aligned} \quad (5)$$

Assume that $x > 1$ (we shall deal with $x = 1$ later), then the second term of δ is

$$\begin{aligned} &\mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon}^m \cup \mathrm{f}) = \\ &\qquad \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{ECR}^*(\langle \ldots, \mathrm{B}_{x-1}, \mathrm{B}_{x+1}, \ldots \rangle) \\ &\quad = \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \\ &\qquad \left[\mathrm{C}_{\mathrm{B}_1} + \sum_{i=2}^{x-1} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) + \frac{\sum_{i=x+1}^{N} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f})}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f})}\right] . \end{aligned}$$

We see that the last term is also represented in Equation 5 and therefore cancels out. We get

$$\begin{aligned}\delta = {} & \mathrm{C}_{\mathrm{B}_1} \cdot [1 - \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f})] + \\ & [1 - \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f})] \cdot \sum_{i=2}^{x-1} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ & + \mathrm{C}_{\mathrm{A}_n} \cdot \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}),\end{aligned}$$

where the last term is a leftover from Equation 5. Using $\mathrm{P}(\neg \mathrm{a} \,|\, \boldsymbol{\varepsilon}) = 1 - \mathrm{P}(\mathrm{a} \,|\, \boldsymbol{\varepsilon})$ and Equation 4 we get

$$\begin{aligned}\delta = {} & \mathrm{C}_{\mathrm{B}_1} \cdot \mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f}) + \\ & \mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f}) \cdot \sum_{i=2}^{x-1} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ & + \mathrm{C}_{\mathrm{A}_n} \cdot \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ \leq {} & \frac{\mathrm{P}(\mathrm{b}_1 \,|\, \mathrm{f})}{\mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f})} \cdot \mathrm{C}_{\mathrm{A}_n} \cdot \mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f}) + \\ & \mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f}) \cdot \sum_{i=2}^{x-1} \frac{\mathrm{P}(\mathrm{b}_i \,|\, \mathrm{f})}{\mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f})} \cdot \mathrm{C}_{\mathrm{A}_n} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ & + \mathrm{C}_{\mathrm{A}_n} \cdot \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ = {} & \mathrm{C}_{\mathrm{A}_n} \cdot \Big[\mathrm{P}(\mathrm{b}_1 \,|\, \mathrm{f}) + \\ & \sum_{i=2}^{x-1} \mathrm{P}(\mathrm{b}_i \,|\, \mathrm{f}) \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ & + \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \Big] \qquad (6) \\ = {} & \mathrm{C}_{\mathrm{A}_n} \cdot \Big[1 - \mathrm{P}(\neg \mathrm{b}_1 \,|\, \mathrm{f}) + \\ & (1 - \mathrm{P}(\neg \mathrm{b}_2 \,|\, \mathrm{f})) \cdot \mathrm{P}(\neg \mathrm{b}_1 \,|\, \mathrm{f}) \\ & + \cdots + \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \Big] \\ = {} & \mathrm{C}_{\mathrm{A}_n} \cdot \Big[1 - \mathrm{P}(\neg \mathrm{b}_2 \,|\, \mathrm{f}) \cdot \mathrm{P}(\neg \mathrm{b}_1 \,|\, \mathrm{f}) \\ & + \cdots + \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \Big] \\ = {} & \mathrm{C}_{\mathrm{A}_n} \cdot 1,\end{aligned}$$

as required. This is not surprising if we look at the expression inside the parenthesis of Equation 6: the corresponding events are "B_1 fixes f, B_2 fixes f if B_1 did not fix f" etc. up to "none of the actions fixed f". These events form a sample space.

When $x = 1$, then $\delta = \mathrm{C}_{\mathrm{A}_n} - \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{C}_{\mathrm{A}_n}$, so in all cases $\delta \leq \mathrm{C}_{\mathrm{A}_n}$ which completes the proof. □

Remark. It is quite straightforward to show that $\underline{\mathrm{ECR}}(\cdot)$ is not monotone when the model includes questions.

3 Pruning based on efficiency and ECR

We recently investigated the effect of a pruning method based on efficiency (Ottosen and Jensen, 2008). By considering two adjacent actions in an optimal troubleshooting sequence, the following has been proved about the efficiency (Jensen et al., 2001):

Theorem 3. *Let* $\mathrm{s} = \langle \mathrm{A}_1, \ldots, \mathrm{A}_n \rangle$ *be an optimal sequence of actions with independent costs. Then it must hold that*

$$\mathrm{ef}(\mathrm{A}_i \,|\, \boldsymbol{\varepsilon}^{i-1}) \quad \geq \quad \mathrm{ef}(\mathrm{A}_{i+1} \,|\, \boldsymbol{\varepsilon}^{i-1}),$$

for all $i \in \{1, \ldots, n-1\}$.

In Figure 2 it is illustrated how the theorem can be used for pruning. If we have the order $\mathrm{ef}(\mathrm{A}_1) > \mathrm{ef}(\mathrm{A}_2) > \mathrm{ef}(\mathrm{A}_3)$ at the root, we know that A_3 should never be the first action. Furthermore, after performing A_2, we know that A_1 should never be the second action. We call this *efficiency-based pruning*.

In summary, the theorem was very easy to exploit during the expansion of a node by keeping the actions sorted with respect to efficiency and by passing that information in the parent node. However, the results where a bit disappointing since it only gave a speed-up of a factor of 2-4.

We have since then tried to extend the idea by considering three adjacent actions instead of two. We call this *ECR-based pruning*. Figure 2 shows an overview of the pruning process. If we consider an arbitrary subset of three actions A_1, A_2, and A_3, we would normally need to compare six different sequences. However, if we have

calculated the efficiencies of the three actions at the local root node with evidence $\boldsymbol{\varepsilon}$, Theorem 3 leaves us with only three possible candidates. After the three sequences are expanded, the paths are coalesced into a single node in the search graph.

Now imagine that A* is about to expand A_3 in the sequence $\langle A_1, A_2, A_3\rangle$. We determine if the current node expansion is optimal by comparing it with the ECR of the sequence $\langle A_2, A_3, A_1\rangle$. (There is no need for comparing $\langle A_1, A_2, A_3\rangle$ with $\langle A_1, A_3, A_2\rangle$ since Theorem 3 has pruned the latter.) If we expand the sequence $\langle A_2, A_3, A_1\rangle$ first, the analysis is similar and we compare with the best of the two other sequences (again, the best sequence is found by applying Theorem 3).

There is no way to avoid calculating the full ECR of both sequences, and we have to traverse the search graph down to the root and up to the first node of the second path. Furthermore, this traversal means that we have to store child pointers in all nodes, and we also need to keep all expanded nodes in memory. This more than doubles the memory requirement.

In conclusion the method turned out to slow down A*. In a model with 19 action and 19 faults, Theorem 3 pruned 989k nodes whereas the ECR-based pruning prevented a mere 3556 expansions out of about 41k possible.

Theorem 2 also explains why the effect of the pruning is so small: if A* is guided by a monotone heuristics and expands a node, then the optimal path to that node has been found (Hart et al., 1968). This means that the sub-trees outlined with dotted edges in Figure 2 are never explored unless we really need to explore them. Should we discover a non-optimal sequence first, that path is not explored further until coalescing happens.

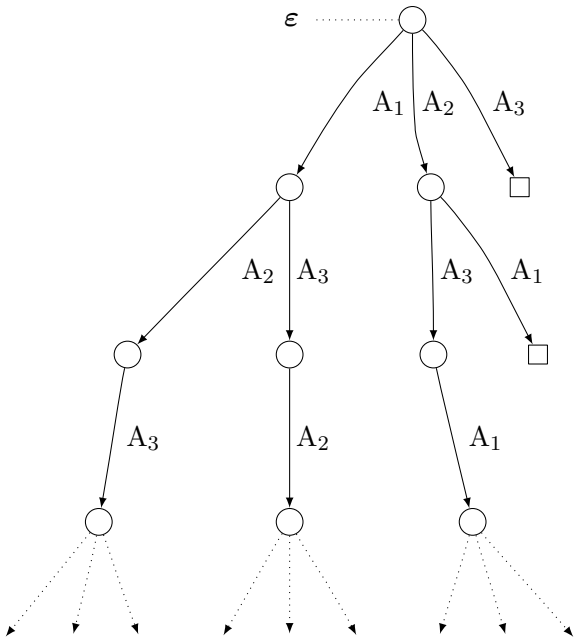


Figure 2: An overview of the pruning process for any subset of three actions. At the root of the subtree we have evidence $\boldsymbol{\varepsilon}$ and the actions are sorted with respect to efficiency, and we have $\mathrm{ef}(A_1|\boldsymbol{\varepsilon}) > \mathrm{ef}(A_2|\boldsymbol{\varepsilon}) > \mathrm{ef}(A_3|\boldsymbol{\varepsilon})$. Theorem 3 implies that we can prune the nodes ending in a square, and so we are left with only three possible sequences ($\langle A_1, A_2, A_3\rangle$, $\langle A_1, A_3, A_2\rangle$, and $\langle A_2, A_3, A_1\rangle$). After A* has discovered the last node in these three sequences, the three paths are subject to coalescing.

4 An A* hybrid approach

Troubleshooting with dependent actions was proved NP-hard by reduction from the exact cover by 3-sets problem (Vomlelová, 2003). Therefore, troubleshooting in models where each action can repair three or more faults is NP-hard. However, A* still has difficulties in finding a solution for models with much lower average dependency (that is, the average size of $\mathrm{fa}(\cdot)$ over all actions). Remarkably enough, we found that when the average dependency went down from 3 towards 1, then the running time increased many times. In the following we describe a hybrid method for models with an average dependency well below 3.

Figure 3 shows an example of the search tree explored by A* in our hybrid approach. Near the root node, A* is often forced to create a branch for each successor node. However, as we get closer to the goal nodes, branching is more likely to be avoided. The branching can be avoided

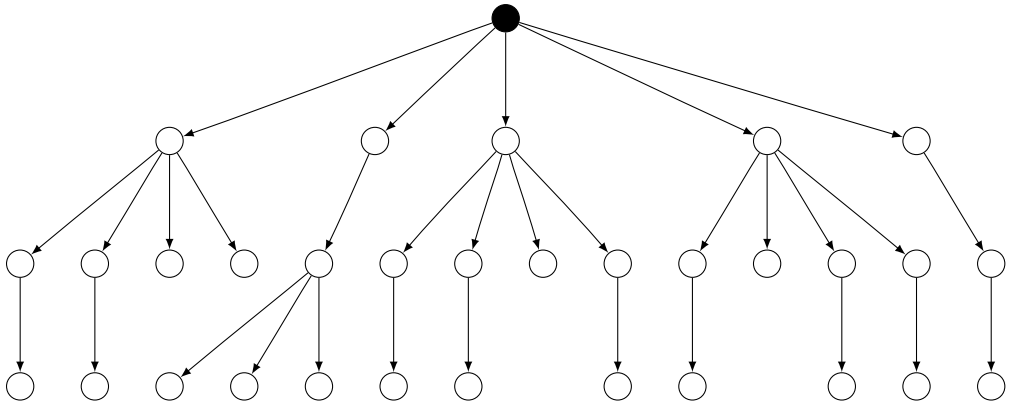


Figure 3: An example of what the search tree looks like in our hybrid approach. For some nodes, the normal A* branching is avoided, and near goal nodes this branching is almost avoided for all nodes. We can see that it might happen that the algorithm has to investigate all successors of a node even though the path down to that node was explored without branching.

because we are able to determine the optimal next step of the remaining sequence (see below).

This leads us to the following definitions:

Definition 6. A *dependency graph* for a troubleshooting model given evidence $\boldsymbol{\varepsilon}$ is the undirected graph with a vertex for each action $\mathrm{A} \in \mathcal{A}(\boldsymbol{\varepsilon})$ and an edge between two vertices A_1 and A_2 if $\mathrm{fa}(\mathrm{A}_1 \,|\, \boldsymbol{\varepsilon}) \cap \mathrm{fa}(\mathrm{A}_2 \,|\, \boldsymbol{\varepsilon}) \neq \emptyset$.

Definition 7. A *dependency set* for a troubleshooting model given evidence $\boldsymbol{\varepsilon}$ is a connectivity component in the dependency graph given $\boldsymbol{\varepsilon}$.

Definition 8. A *dependency set leader* for a troubleshooting model given evidence $\boldsymbol{\varepsilon}$ is the first action of an optimal sequence in a dependency set given $\boldsymbol{\varepsilon}$.

Dependency sets are important because the order of actions in the same dependency set does not change when actions outside the set are performed. This property has been exploited in the following theorem (Koca and Bilgiç, 2004):

Theorem 4. *Suppose we are able to calculate the dependency set leaders. Then the globally optimal sequence is given by the following algorithm:*

1. *Construct the dependency sets and retrieve the set leaders.*
2. *Calculate* $\mathrm{ef}(\cdot)$ *for all set leaders.*
3. *Select the set leader with the highest* $\mathrm{ef}(\cdot)$ *and perform it.*
4. *If it fails, update the probabilities, and continue in step (2).*

Our hybrid approach then simply works by finding the optimal sequence in dependency sets of a fairly small size. For this work we have restricted us to sets of a size < 4. At any point before expanding a node, if the most efficient action belongs to a dependency set of such a small size, we find the first action in that dependency set. If the dependency set consists of one or two actions, this calculation is trivial. If the dependency set has three actions, we find the first by comparing the three candidate sequences as we discussed in Section 3. Otherwise we simply expand the node as usual by inspecting all successors.

Table 4 shows the results of three versions of A*. We can see that the hybrid approach is somewhat slower for models with an average dependency between 2 and 3. This is because the hybrid approach spends time investigating the size of the dependency set of the most efficient action, but it rarely gets to exploit the benefits of a small dependency set. For an average dependency between 2.1 and 1.6 the hybrid approach becomes superior, and below 1.6 it becomes very fast.

Table 1: Results for the hybrid approach in models with 20 actions and 20 faults. The average dependency ranges between 3 and 1, and each action is usually associated with 1 to 3 faults. The time is measured in seconds. "A* " is A* with coalescing, "pruning-A* " is "A* " plus efficiency-based pruning and "hybrid-A* " is the hybrid approach based on "pruning-A* ". Already at an average dependency around 2.1 we see that the hybrid method wins.

Method	A*	pruning-A*	hybrid-A*
Dep.		*Time*	
3.0	33.56	11.48	12.27
2.9	42.11	12.42	21.97
2.8	62.14	15.08	27.52
2.7	45.03	14.38	21.61
2.6	29.86	10.20	12.39
2.5	86.52	22.20	29.61
2.4	31.55	12.39	12.00
2.3	65.19	19.11	21.28
2.2	80.56	21.28	29.38
2.1	50.28	18.72	9.78
2.0	83.75	27.70	20.05
1.9	62.88	16.77	10.64
1.8	127.59	35.09	18.72
1.7	102.17	25.36	14.42
1.6	133.17	39.14	25.41
1.5	122.08	27.59	0.92
1.4	164.84	41.16	4.25
1.3	139.89	39.44	0.13
1.2	168.42	38.13	0.00
1.1	160.42	39.42	0.00
1.0	159.95	38.08	0.00

5 Discussion

We originally investigated Theorem 2 because monotonicity plays an important role in promising bidirectional A* methods (Kaindl and Kainz, 1997). However, a bidirectional A* for troubleshooting is very difficult because there is no apparent way to start a search from the goal nodes using $\underline{\mathrm{ECR}}(\cdot)$.

The hybrid A* approach seems very promising. We still need to determine how large a dependency set that it pays off to solve. We expect that it will be most beneficial to solve small dependency sets by brute-force whereas dependency sets of medium size can be solved by a recursive call to hybrid-A* .

Acknowledgements

We would like to thank the reviewers for their valuable and detailed feedback.

References

Rina Dechter and Judea Pearl. 1985. Generalized best-first search strategies and the optimality af a*. *J. ACM*, 32(3):505–536.

P. E. Hart, N. J. Nilsson, and B. Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, SSC-4(2):100–7.

Finn V. Jensen, Uffe Kjærulff, Brian Kristiansen, Claus Skaanning, Jiri Vomlel, and Marta Vomlelová. 2001. The sacso methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15:321–333.

J. Kadane and H. Simon. 1977. Optimal strategies for a class of constrained sequential problems. *The Annals of Statistics*, 5:237–255.

Hermann Kaindl and Gerhard Kainz. 1997. Bidirectional heuristic search reconsidered. *Journal of Artificial Intelligence Research*, 7:283–317.

Eylem Koca and Taner Bilgiç. 2004. A troubleshooting approach with dependent actions. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI 2004: 16th European Conference on Artificial Intelligence*, pages 1043–1044. IOS Press.

Thorsten J. Ottosen and Finn V. Jensen. 2008. Better safe than sorry—optimal troubleshooting through A* search with efficiency-based pruning. In *Proceedings of the Tenth Scandinavian Conference on Artificial Intelligence*, pages 92–97. IOS Press.

M. Vomlelová and J. Vomlel. 2003. Troubleshooting: Np-hardness and solution methods. *Soft Computing Journal, Volume 7, Number 5*, pages 357–368.

Marta Vomlelová. 2003. Complexity of decision-theoretic troubleshooting. *Int. J. Intell. Syst.*, 18(2):267–277.

Discrimination and its sensitivity in probabilistic networks

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

A probabilistic network built for an application domain often has a single output variable of interest, for which either the posterior probability of one of its values or its most likely value is reported and used for subsequent decision making. For our domain of application, however, we are interested primarily in how well the network distinguishes between various compound output values of interest for different diagnostic variables. To capture this, we introduce a concept of discrimination, and illustrate a measure to this end, based upon joint posterior probabilities. In addition, we address the sensitivity of discrimination to inaccuracies in a network's parameters and show that standard sensitivity functions suffice for studying the effects of such inaccuracies.

1 Introduction

A probabilistic network designed for diagnostic support in an application domain often has a single output variable of interest, capturing the possible diagnoses. Our application domain of classical swine fever, however, aims at multiple-disorder diagnosis. To this end, we have two output variables of interest: a main diagnostic variable to detect classical swine fever, and a secondary variable to capture primary other infections. Although outbreaks of classical swine fever occur seldomly, it is a very serious infectious disease which warrants early detection to prevent rapid spreading. Early detection, however, is hampered by close resemblance of the early symptoms of the disease to those of common infections, and by the simultaneous presence of such infections. A model for early detection of classical swine fever, therefore, needs to be able to distinguish between classical swine fever in an early stage and a primary other infection. Moreover, it should be capable of diagnosing classical swine fever in combination with common infections.

In order to determine how well a probabilistic network can distinguish between different diagnoses in an individual case, it does not always suffice to consider the most likely value of a variable of interest, or its posterior distribution, especially when more than one diagnostic variable is concerned. Therefore, we introduce the concept of *evidence-specific discrimination* between values of one or more variables. Various measures involving posterior probabilities for the diagnoses of interest can be used to capture such discrimination. In this paper we illustrate the concept of discrimination by defining the absolute difference between posterior probabilities as a simple discrimination measure: the further these probabilities are apart, the better the network discriminates between the associated diagnoses.

We note that the term discrimination is somewhat overloaded; it is, for example, often used in the context of classification problems: "can our model discriminate between pigs that have classical swine fever and pigs that have not?" This question, although relevant, concerns discrimination between *cases*, and not between diagnoses in an individual case, which is the problem we address here.

Posterior probabilities can be highly sensitive to changes in a probabilistic network's numerical parameters (Van der Gaag & Renooij, 2001). As the parameters are generally estimated from (incomplete) data or assessed by human experts in the domain of application, they are inevitably inaccurate. To study the robustness of discrimination to parameter inaccuracies, we can study the sensitivity of the output probabilities involved to parameter changes by means of a sensitivity analysis. To

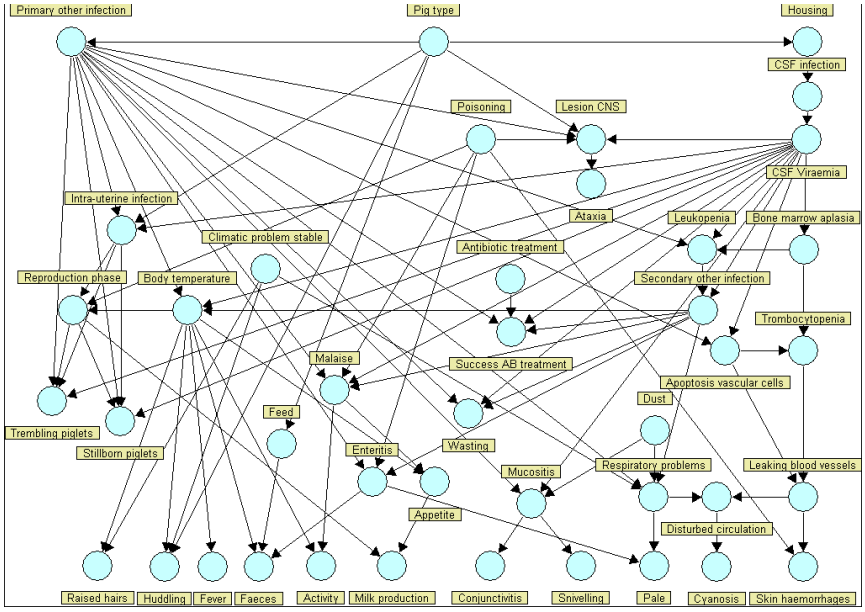


Figure 1: The network for early detection of classical swine fever (csf).

this end, we show how to derive a function that captures the *sensitivity of discrimination* to parameter changes. In addition, we demonstrate that we can efficiently compute a sensitivity function for joint posterior probabilities, required in order to study the dynamics, and therefore robustness, of discrimination between values of two or more variables.

The paper is organised as follows. In Section 2 we describe an application which motivates the need for a concept of discrimination and introduce a preliminary measure to this end. In Section 3, we establish functions that allow for studying the robustness of discrimination to parameter inaccuracies. The paper ends with conclusions and directions for further research in Section 4.

2 The Concept of Discrimination

Classical swine fever (csf) is a serious infectious disease and, although outbreaks occur seldomly, its rapid spreading warrants early detection. Classical swine fever is hard to diagnose in an early stage, due to the high variation in its associated clinical patterns, which strongly resemble those of common other infectious diseases. To support the early detection of csf, a probabilistic network is being developed (Geenen, Elbers, Van der Gaag & Loeffen, 2006). The network, shown in Figure 1, currently includes 82 directed edges, 2113 conditional probabilities, and 41 variables of which 24 can be observed upon clinical investigation. The variables capture processes in the underlying pathogenesis, risk factors, relevant clinical signs, and alternative explanations for these signs.

The main diagnostic variable CSF Viraemia in the network models the presence or absence of csf in an individual pig. The extremely low prior for the presence of csf (0.0000016), in combination with the common occurrence of other infections resembling csf, both in pigs with and without csf, makes that these diseases cannot all be modelled in a single variable: csf would never be diagnosed. A secondary diagnostic variable in the network therefore models primary other infections as possible alternative explanations of a pig's symptoms. As a result, for a given pig, not only the network's prediction of the probability of csf is of interest, but our primary interest is to know how well the network distinguishes csf in an early stage, with or without another infection being present, from just a primary other infection.

Known concepts as the most likely value of a variable of interest, or its posterior distribution, do not always suffice to determine how well a probabilistic network can distinguish between different diagnoses in an individual case, especially when more than one diagnostic variable is concerned. To

capture this, we therefore introduce the novel concept of *evidence-specific discrimination* between two combinations of values for one or more variables. To measure discrimination, we use a function of the posterior probabilities of the (compound) values of interest. This *measure of discrimination* preferably has the property of obtaining a maximum value when one of the posteriors equals zero and the other 1: it is then easy to discriminate between the two associated diagnoses; likewise, the measure should obtain a minimum value when the posteriors are equal. Possible measures of discrimination can be based on (odds) ratios, or more complex functions. In this paper, for the purpose of illustration, we use a simple and straightforward measure of discrimination, defined below. In the remainder of this paper, we will write $\Pr(a \mid e)$, to denote an output probability under study, where a is a specific value assignment to one or more variables A of interest and e denotes the available evidence.

Definition 1. Let $\Pr(a \mid e)$ and $\Pr(b \mid e)$ be two output probabilities of interest. The amount of *discrimination* of the network between a and b in the context of evidence e, written $d(a\,; b \mid e)$, equals $\mid \Pr(a \,|\, e) - \Pr(b \,|\, e) \mid$.

The above measure $d(a\,; b \mid e)$ takes on values between zero and 1, with larger values indicating a larger amount of discrimination. This specific measure also has the benefit of being symmetric in its arguments, that is, $d(a; b \,|\, e) = d(b; a \,|\, e)$.

Example 1. Discrimination can be measured between different values of the same variable. For example, discrimination between a gastro-intestinal infection (value gi of variable POI, modelling primary other infections) and an airway infection (value ai of variable POI) in pig 14 is given by

$$\begin{aligned} d(gi; ai \,|\, 14) &= \mid \Pr(gi \,|\, 14) - \Pr(ai \,|\, 14) \mid \\ &= 0.54 - 0.10 = 0.44 \end{aligned}$$

indicating that the network can easily distinguish between these two types of infection in this pig. Discrimination can also be studied for values of different variables. For example, discrimination between classical swine fever (value csf of variable CSF) and a gastro-intestinal infection in pig 169 is given by $d(csf; gi \,|\, 169) = \mid \Pr(csf \,|\, 169) - \Pr(gi \mid 169) \mid = 0.20 - 0.13 = 0.07$, indicating that the network has some difficulty distinguishing csf from a common infection in this pig. Discrimination can even be studied for value assignments to more than one variable. For example, discrimination between the presence and absence of classical swine fever in combination with an airway infection in pig 304: $d(csf, ai\,; \neg csf, ai \mid 304) = \mid \Pr(csf, ai \mid 304) - \Pr(\neg csf, ai \,|\, 304) \mid = \mid 0.01 - 0.15 \mid = 0.14$; this indicates that the network is capable of diagnosing csf in combination with another infection in this pig. □

3 Robustness of Discrimination

Sensitivity analysis is a powerful tool for studying the robustness of a probabilistic network's output probabilities to inaccuracies in the network parameters. Since discrimination is defined in terms of output probabilities, its robustness to parameter changes is a relevant matter, and can be studied by means of the functions that result from a sensitivity analysis. We now review some known properties of such sensitivity functions.

3.1 Sensitivity Functions

Sensitivity analysis of a probabilistic network amounts to establishing, for each of the network's numerical parameters, the *sensitivity function* that expresses an output probability of interest in terms of that parameter. Let $x = p(b \mid \pi)$ be a parameter under study, where b is a value of some variable B and π is a combination of values for B's parents. We now use $f_a^e(x)$ to denote the sensitivity function that expresses the output probability $\Pr(a \mid e)$ in terms of the parameter x.

Any one-way sensitivity function $f_a^e(x)$ is a quotient of two linear functions in the parameter x under study (Castillo, Gutiérrez & Hadi, 1997; Coupé & Van der Gaag, 2002). More formally, the function takes the form

$$f_a^e(x) = \frac{\Pr(a,\, e)(x)}{\Pr(e)(x)} = \frac{c_1 \cdot x + c_2}{c_3 \cdot x + c_4}$$

where the constants c_j, $j = 1, \ldots, 4$, are built from the assessments for the parameters that are not being varied[1]. Efficient algorithms exist to compute

[1] We assume that the parameters pertaining to the same conditional distribution as the parameter under study are co-varied proportionally (Kjærulff & Van der Gaag, 2000).

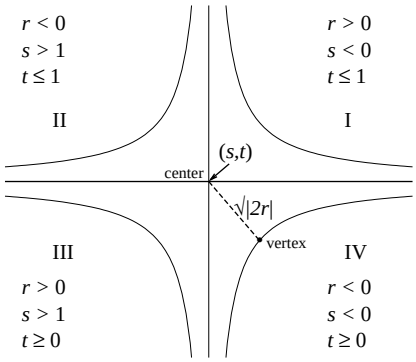


Figure 2: Two hyperbolas with their branches and associated constants (the constraints on s and t are specific for sensitivity functions).

the constants of any sensitivity function relating a (posterior) probability for a value of a single output variable to a network parameter (Coupé & Van der Gaag, 2002; Kjærulff & Van der Gaag, 2000).

The sensitivity function $f_a^e(x)$ can take one of three general forms. The function is *linear* for prior probabilities of interest, or if $\Pr(e)$ is unaffected by the parameter variation ($c_3 = 0$); if $c_4 = 0$, then $c_2 = 0$ and the function reduces to a *constant*. In all other cases the function is a fragment of a *rectangular hyperbola*, which takes the general form

$$f(x) = \frac{r}{x - s} + t$$

where, for a sensitivity function with $c_1, \ldots, c_4$ as before, $s = -c_4/c_3$, $t = c_1/c_3$, and $r = (c_2/c_3) + s \cdot t$. In the remainder of the paper, we assume any sensitivity function to be hyperbolic.

Figure 2 illustrates that a rectangular hyperbola in general has two branches, and two asymptotes defining its center (s, t). We observe that a sensitivity function is defined by $0 \leq x, f(x) \leq 1$; the two-dimensional space of feasible points thus defined, is termed the *unit window*. Since a sensitivity function moreover should be continuous for $x \in [0, 1]$, its vertical asymptote necessarily lies outside the unit window. A hyperbolic sensitivity function therefore is a fragment of a single hyperbola branch.

3.2 Discrimination Dynamics: Simple Values

The robustness of a network's discrimination between a and b, in the context of evidence e, to changes in a parameter x, can be captured by considering $d(a; b \,|\, e)$ as a function of x. In this section we assume that a and b are *simple* values, that is values for a single variable A and a single variable B; the case where a and b are *compound* values is considered in Section 3.3. In this paper we assume that $d(a; b \mid e)$ itself is a function involving simple operators as the sum, the difference, and/or the ratio of posterior probabilities. We will demonstrate, for our choice of measure, that *discrimination sensitivity* $d(a; b|e)(x)$ can now again be described in terms of a rectangular hyperbola.

Proposition 1. *Let $f_a^e(x) = r_a/(x - s) + t_a$ and $f_b^e(x) = r_b/(x-s)+t_b$ be two sensitivity functions. Then*

$$f_a^e(x) - f_b^e(x) = \frac{(r_a - r_b)}{x - s} + (t_a - t_b)$$

The above immediately follows from having the same constant s in both sensitivity functions, which is justified by the following lemma.

Lemma 1. *For a fixed parameter x and evidence e, all sensitivity functions $f_A^e(x)$ for any variable A have the same vertical asymptote.*

Recall that the constant s for a sensitivity function $f_a^e(x)$ equals $s = -c_4/c_3$, where $c_3 \cdot x + c_4 = \Pr(e)(x)$. Given a parameter x, constant s therefore relates to just the available evidence and is independent of the output variable of interest.

Although the difference function from Proposition 1 again is a fragment of one hyperbola branch for $x \in [0, 1]$, it will in general not be a sensitivity function since it can be negative on $[0, 1]$; for our choice of measure, $d(a\,; b \mid e)(x)$ is the absolute value of this difference. For a fixed parameter x and evidence e, establishing the constants of all sensitivity functions $f_A^e(x)$ for any single variable A, rather than for one specific A, comes at no additional computational expense. Establishing $d(a; b|e)(x)$ hence requires no additional network propagations.

The function $d(a; b \mid e)(x)$ now details how discrimination is affected by parameter variation. Discrimination is robust to parameter inaccuracies if its change upon varying a parameter is limited. To assess robustness, we now define different intervals of parameter values having different effects on discrimination.

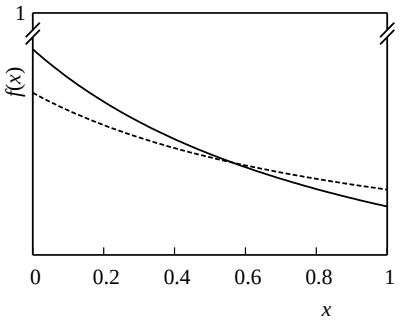


Figure 3: Example sensitivity functions with $s = -1$, $x_{\text{int}} = 0.56$ and $x_{\text{max}} = 0$.

Since $d(a\,;b \mid e)(x)$ is based on two sensitivity functions, which are continuous and monotone for $x \in [0,1]$, we have that maximum discrimination is found on the boundaries of the unit window, that is, for either $x = 0$ or $x = 1$. The value of parameter x where discrimination is maximised will be denoted by x_{max}:

$$x_{\text{max}} = \operatorname*{argmax}_{x \in [0,1]} d(a;b \,|\, e)(x) \in \{0,1\}$$

If the two sensitivity functions $f^e_a(x)$ and $f^e_b(x)$ for the posterior probabilities under consideration intersect within the unit window, such as in Figure 3, then minimum discrimination is attained at this intersection point. Assuming that the two hyperbolas are truly different functions, that is $f^e_a(x) \neq f^e_b(x)$, they intersect for at most one value of x, denoted x_{int}. For our choice of discrimination measure this minimum value equals zero and is attained at

$$d(a;b \,|\, e)(x_{\text{int}}) = 0 \iff x_{\text{int}} = \frac{r_a - r_b}{t_b - t_a} + s$$

If $x_{\text{int}} \in \langle 0,1 \rangle$, then parameter values on opposite sides of x_{int} will result in the same amount of discrimination between the values a and b under consideration (see Figure 4). Let x_{sim} denote the value of x for which discrimination equals the original amount of discrimination between a and b in context e, that is

$$d(a;b \,|\, e)(x_{\text{sim}}) = d(a;b \,|\, e)(x_0),$$

where x_0 is the original value of the parameter x under consideration. For our example measure, the

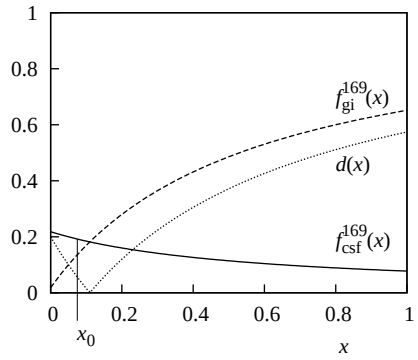


Figure 4: Discrimination $d(csf\,;gi \mid 169)(x)$ for a parameter x with $x_0 = 0.075$; $x_{\text{int}} = 0.11$ and $x_{\text{sim}} = 0.16$.

value of x_{sim} can be easily established from $d_0 = f^e_a(x_0) - f^e_b(x_0)$:

$$x_{\text{sim}} = \frac{r_a - r_b}{t_b - t_a - d_0} + s$$

We now note that x_0 and x_{sim} necessarily lie on opposite sides of x_{int} if the latter two lie within the unit window. As a result, for x-values between x_0 and x_{sim}, discrimination will become smaller than its original value at x_0; we will then say that discrimination decreases, even though it is not a decreasing function of x; for x-values outside the interval bounded by x_0 and x_{sim} discrimination increases. If $x_{\text{sim}} \notin [0,1]$, for example as in Figure 3 with $x_0 = 0.10$ and $x_{\text{sim}} = 1.70$, then, necessarily, x_{max} lies on the same side of x_{int} as x_0, so variation of x from x_0 towards x_{max} increases discrimination, whereas discrimination will become less when x is varied in the opposite direction.

The intersection point of the two hyperbolas does not necessarily lie within the unit window (see for example Figure 5). If the intersection point lies outside the unit window, or if the hyperbola branches do not intersect at all, then discrimination $d(a\,;b \mid e)(x)$ is monotone for $x \in [0,1]$, obtaining its minimum value at $1 - x_{\text{max}}$.

We now have all the ingredients to describe the effect of parameter variation on discrimination.

Proposition 2. *Let $f^e_a(x)$, $f^e_b(x)$, x_{max}, x_{int}, and x_{sim} be as before. Then the network's discrimination $d(a\,;b \mid e)$ between a and b in the context of evidence e changes as follows, upon varying parameter x:*

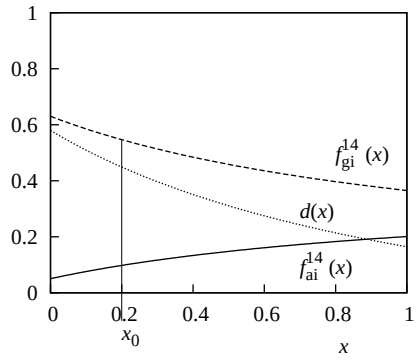


Figure 5: Discrimination $d(gi\,;\,ai \mid 14)(x)$ for a parameter x with $x_0 = 0.20$; $x_{\text{int}} = 2.08$, $x_{\text{sim}} = -1.17$, and $x_{\max} = 0$.

- *if* $x_{\text{int}} \notin [0,1]$ *or* $x_{\text{sim}} \notin [0,1]$*, then discrimination is non-decreasing if* x *is varied towards* $x_{\max}$*, and non-increasing otherwise.*
- *if* $x_{\text{int}} \in [0,1]$ *and* $x_{\text{sim}} \in [0,1]$*, then discrimination is non-decreasing if* x *is varied to* $x \leq \gamma$ *or to* $x \geq \delta$*, where* $[\gamma, \delta] = [\min\{x_0, x_{\text{sim}}\}, \max\{x_0, x_{\text{sim}}\}]$*, and non-increasing otherwise.*

Example 2. Reconsider the network for early detection of csf and its discrimination between csf and a gastro-intestinal infection for pig 169, $d(csf\,;\,gi \mid 169)$. Discrimination between these two values as a function of a parameter x pertaining to the success of treatment with antibiotics ($x_0 = 0.075$), is captured by the following two sensitivity functions:

$$f^{169}_{csf}(x) = \frac{0.12}{x + 0.55} \text{ and } f^{169}_{gi}(x) = \frac{-0.54}{x + 0.55} + 1$$

and shown in Figure 4. From these functions we find that $x_{\text{int}} = 0.11$ and $x_{\text{sim}} = 0.16$. As a result, if a more accurate assessment of the parameter turns out smaller than 0.075, then the network will be able to better discriminate between the two infections in this pig; if a more accurate assessment is larger, but not as large as 0.16, then discrimination becomes worse. Similarly, discrimination $d(gi\,;\,ai \mid 14)$ between a gastro-intestinal and an airway infection in pig 14, as a function of a parameter x pertaining to faeces samples ($x_0 = 0.20$), is captured by the following two sensitivity functions:

$$f^{14}_{gi}(x) = \frac{0.69}{x + 1.19} + 0.05, \text{ and}$$

$$f^{14}_{ai}(x) = \frac{-0.39}{x + 1.19} + 0.38$$

shown in Figure 5. From the formulas we find that $x_{\text{int}} = 2.08$ and $x_{\text{sim}} = -1.17$. We conclude that discrimination decreases for any parameter value larger than 0.20, and increases otherwise. □

For our example measure we have studied the difference between two sensitivity functions, each describing a posterior probability for a simple output value as a function of a network parameter. We note that the difference between two such posterior probabilities in relation to changes in a network parameter has been studied before by Chan & Darwiche (2002) in the context of parameter tuning. They demonstrated that parameter values which enforce a constraint on the difference, or on the ratio, of two posterior probabilities can be computed from partial derivatives established from the network without explicitly determining a sensitivity function. Establishing the constants of the sensitivity function, however, is just as efficient and has the benefit of providing insight in the effects of arbitrary parameter changes on an output of interest.

3.3 Discrimination Dynamics: Compound Values

In this section, we extend the results on the robustness to parameter inaccuracies of discrimination between simple values, to apply to compound value combinations for two or more variables. This type of robustness is relevant in case we need to distinguish between diagnoses for multiple disorders, modelled in separate diagnostic variables. In practice, the number of variables under consideration should typically be small, for computational reasons as well as for interpretability. Although results in this section apply to compound values for any number of variables, we limit the discussion to only two.

We consider the variables A and B and the posterior probability $\Pr(a,\, b \mid e)$ of the compound value ab. A sensitivity function $f^e_{ab}(x)$ for ab in the context of evidence e, as a function of x, is readily determined by one of the following two approaches:

I) Extend the network with a new variable Y, with parents A and B and all their compound values as possible values for Y; for the CPT, define $p(y \mid a,\, b) = 1$ iff $y \equiv ab$, and

$p(y \mid a, b) = 0$ otherwise. Enter evidence e into the new network and compute $f^e_{Y=ab}(x)$.

II) Enter evidence e into the original network to compute $f^e_b(x)$, then enter additional evidence b to compute $f^{be}_a(x)$. Finally, multiply the two functions:

$$\begin{aligned} f^e_{ab}(x) &= f^{be}_a(x) \cdot f^e_b(x) \\ &= \left(\frac{r_a}{x - s_{be}} + t_a\right) \cdot \left(\frac{r_b}{x - s_e} + t_b\right) \end{aligned}$$

The first approach requires less propagations, but in a more complex network, and establishes the sensitivity functions for all compound values of the variables under consideration. The second approach leaves the network as-is and provides all information for establishing the sensitivity functions $f^e_{Ab}(x)$ for all values of all variables A in the network. The multiplication step is simplified by the observation that the resulting function is again a sensitivity function and therefore a rectangular hyperbola. This indeed follows after careful inspection of all constants involved.

Proposition 3. *Let $f^{be}_a(x) = r_a/(x - s_{be}) + t_a$ and $f^e_b(x) = r_b/(x - s_e) + t_b$ be two sensitivity functions. Then*

$$f^e_{ab}(x) = \frac{r_a \cdot t_b + (s_e - s_{be}) \cdot t_a \cdot t_b}{x - s_e} + t_a \cdot t_b$$

is the sensitivity function relating the joint probability $\Pr(a, b \mid e)$ to parameter x.

Proof. First we rewrite the formulas for the hyperbolic sensitivity functions in terms of a fraction of linear functions:

$$f^{be}_a(x) = \frac{\Pr(a, b, e)(x)}{\Pr(b, e)(x)} = \frac{c_1 \cdot x + c_2}{c_3 \cdot x + c_4}$$

where $-c_4/c_3 = s_{be}$, $c_1/c_3 = t_a$, and $c_2/c_3 = r_a - s_{be} \cdot t_a$, and

$$f^e_b(x) = \frac{\Pr(b, e)(x)}{\Pr(e)(x)} = \frac{c_3 \cdot x + c_4}{c_5 \cdot x + c_6}$$

where $-c_6/c_5 = s_e$, $c_3/c_5 = t_b$, and $c_4/c_5 = r_b - s_e \cdot t_b$. Now,

$$\begin{aligned} f^e_{ab}(x) &= f^{be}_a(x) \cdot f^e_b(x) = \frac{c_1 \cdot x + c_2}{c_5 \cdot x + c_6} \\ &= \frac{r_{ab}}{x - s_{ab}} + t_{ab} \end{aligned}$$

where

$$\begin{aligned} s_{ab} &= -\frac{c_6}{c_5} = s_e \\ t_{ab} &= \frac{c_1}{c_5} = \frac{c_1}{c_3} \cdot \frac{c_3}{c_5} = t_a \cdot t_b \\ r_{ab} &= \frac{c_2}{c_5} + s_e \cdot t_a \cdot t_b = \frac{c_2}{c_3} \cdot \frac{c_3}{c_5} + s_e \cdot t_a \cdot t_b \\ &= (r_a - s_{be} \cdot t_a) \cdot t_b + s_e \cdot t_a \cdot t_b \end{aligned}$$

□

From the above observations, we have that all properties for sensitivity functions and discrimination derived in the previous section readily apply to the compound values case.

Example 3. Reconsider the network for early detection of csf. We study the network's discrimination, and its robustness, between csf and one of the primary other infections. More specifically, since other infections are quite common in pigs, we are interested in whether or not csf can be distinguished from them. In this example, we focus on the difference between $\Pr(csf, \neg gi \mid 169)$ and $\Pr(\neg csf, gi \mid 169)$ for pig 169. The robustness of discrimination, as a function of a parameter x, can be studied by means of the corresponding sensitivity functions: $|f^{169}_{csf, \neg gi}(x) - f^{169}_{\neg csf, gi}(x)|$. The constants for the rectangular hyperbola representing this difference are found by applying Propositions 3 and 1 to the functions $f^{169, \neg gi}_{csf}(x)$, $f^{169, gi}_{\neg csf}(x)$, and $f^{169}_{gi}(x)$ and exploiting the fact that $f^{169}_{\neg gi}(x) = 1 - f^{169}_{gi}(x)$. From these constants, x_{int} and x_{sim} can be straightforwardly computed.

Examples of the sensitivity functions for the simple output values under consideration and a parameter x pertaining to the success of an antibiotics treatment are given in Figure 6. The sensitivity functions for the compound values of interest for the same x, together with discrimination as a function of x, are shown in Figure 7. Note that Figure 7 gives valuable insight into the dynamics of discrimination between csf and gastro-intestinal infections, which is not obvious from Figure 6: although from Figure 6 we can see that changes in the posterior probability of gi will pull the probabilities for its combination with csf towards the center of the probability range, it is not immediately obvious from this figure that the functions for the compound values will intersect, nor where this will occur. □

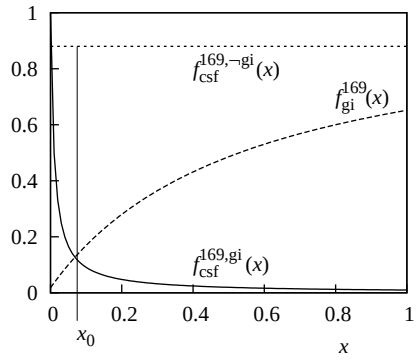


Figure 6: Sensitivity functions for simple values of variables CSF and POI, for a parameter x pertaining to the success of an antibiotics treatment.

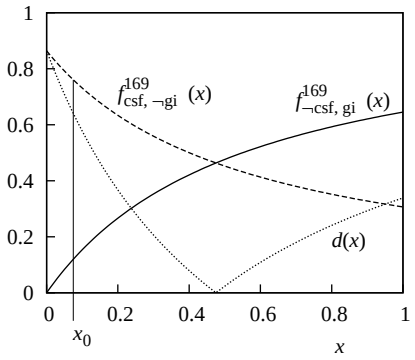


Figure 7: Sensitivity functions for compound values of variables CSF and POI, for a parameter x pertaining to the success of an antibiotics treatment, together with discrimination $d(csf, \neg gi; \neg csf, gi \mid 169)(x)$ for parameter x.

4 Conclusions

In this paper we introduced the concept of evidence-specific discrimination to investigate how well a probabilistic network can distinguish between two or more output values or, more in general, between value combinations for two or more output variables of interest. We illustrated a simple measure of discrimination based on the difference between two posterior probabilities. Subsequently, we demonstrated how sensitivity functions can be employed to study the robustness of discrimination to parameter inaccuracies, even when discrimination concerns compound values rather than simple ones.

Our results on the dynamics of discrimination build to a large extent on the observation that, in the same evidence context, simple operations on hyperbolic sensitivity functions for the same parameter x, again result in a rectangular hyperbola. This entails that more sophisticated discrimination measures, such as for example (odds) ratios or $|f_a^e(x) - f_b^e(x)|/(f_a^e(x) + f_b^e(x))$, can be straightforwardly employed with the techniques presented in this paper. Further research is required to investigate what measure of discrimination is most suitable in what situation, and what amount of discrimination is acceptable or desirable. In addition, we plan on investigating to what extent results that address evidence-dependent bounds on sensitivity functions (Renooij & Van der Gaag, 2005) can be employed to make general statements concerning discrimination involving all network parameters.

References

E. Castillo, J.M. Gutiérrez, A.S. Hadi (1997). Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 27, pp. 412 – 423.

H. Chan, A. Darwiche (2002). When do numbers really matter? *Journal of Artificial Intelligence Research*, vol. 17, pp. 265 – 287.

V.M.H. Coupé, L.C. van der Gaag (2002). Properties of sensitivity analysis of Bayesian belief networks. *Annals of Mathematics and Artificial Intelligence*, vol. 36, pp. 323 – 356.

P.L. Geenen, A.R.W. Elbers, L.C. van der Gaag, W.L.A. Loeffen. Development of a probabilistic network for clinical detection of classical swine fever. *Proceedings of the 11th Symposium of the International Society for Veterinary Epidemiology and Economics*, Cairns, Australia, pp. 667669, 2006.

U. Kjærulff, L.C. van der Gaag (2000). Making sensitivity analysis computationally efficient. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, pp. 317 – 325.

L.C. van der Gaag, S. Renooij (2001). Analysing sensitivity data from probabilistic networks. *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, pp. 530 – 537.

S. Renooij, L.C. van der Gaag (2005). Exploiting evidence-dependent sensitivity bounds. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, AUAI Press, Corvallis, OR, pp. 485-492.

An empirical analysis of loopy belief propagation in three topologies: grids, small-world networks and random graphs

R. Santana, A. Mendiburu and J. A. Lozano
Intelligent Systems Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country
Paseo Manuel de Lardizábal, 20018 San Sebastian - Donostia, Spain.
rsantana,alexander.mendiburu,ja.lozano@ehu.es

Abstract

Recently, much research has been devoted to the study of loopy belief propagation algorithm. However, little attention has been paid to the change of its behavior in relation with the problem graph topology. In this paper we empirically study the behavior of loopy belief propagation on different network topologies which include grids, small-world networks and random graphs. In our experiments, several descriptors of the algorithm are collected in order to analyze its behavior. We show that the performance of the algorithm is highly sensitive to changes in the topologies. Furthermore, evidence is given showing that the addition of shortcuts to grids can determine important changes in the dynamics of the algorithm.

1 Introduction

Loopy belief propagation (LBP) (Pearl, 1988) is a very efficient message-passing algorithm that has been applied to a variety of inference and optimization problems. One of the factors that influences the accuracy and efficiency of LBP and other message-passing algorithms is the underlying graphical structure (or topology) of the graphical model where the inference algorithm is applied. Although it is known that the existence of cycles in the graph has an impact on the behavior of LBP, little attention has been given to the study of the relationship between other characteristics of the graph topology and the LBP behavior. Moreover, few papers consider possible ways of using the graph topology to adapt the LBP implementation.

A paper that can be considered an exception to this trend is (Ohkubo et al., 2005). It describes a modification of LBP that takes into account the information about the topological heterogeneity of the complex networks where it is applied. It turns out that by modifying the asynchronous message-passing schedule according to the degree of the network vertices it is possible to increase the efficiency of LBP. The use of topological information in the scheduling of the messages could also be seen as another way of conceiving *informed scheduling schemes* of which residual belief propagation (Elidan et al., 2006) is perhaps the best known example.

Natural candidates for analyzing the influence of the graph topology on LBP are complex networks, whose topological characteristics are midway between those of regular lattice or grids and random graphs. The recent surge on the study of complex networks (Watts and Strogatz, 1998) is mainly due to their suitability as a framework for the study of complex systems (Dorogovtsev et al., 2007).

One example of complex networks are small-world networks which simultaneously hold some particular characteristics of regular lattices, such as the existence of local clustering between neighboring vertices, with other attributes characteristic of random networks, such as the short average distance between pairs of vertices. This combination of attributes makes them more ap-

propriate than grids and random graphs to represent interaction networks of real phenomena such as neuronal, social and genetic networks (Watts and Strogatz, 1998), electronic circuits (Ferrer i Cancho et al., 2001), etc.

In this paper, we analyze two kinds of problems related with the use of LBP on networks of different topology:

1. We consider the change in the dynamics of LBP when there is a variation in the graphical structure. In particular, we investigate whether there are significant differences in the behavior of LBP for problems defined in grids, small-world and random graphs.

2. We also investigate whether it is possible to influence the performance and behavior of LBP by modifying the graphical structure without changing the function values.

The paper is organized as follows: In the following section the main concepts related to the class of complex networks and factor graphs are introduced. Section 3 briefly reviews the LBP algorithm and in Section 4 the main characteristics of our implementation, FlexLBP, are explained. In Section 5, the use of LBP across the different classes of chosen topologies is analyzed. Section 6 presents the experiments and analyzes their results. The paper ends in Section 7 where the conclusions and topics for future work are presented.

2 Small-world networks and factor graphs

2.1 Small-world networks

Let $G = (V, E)$ be an undirected graph, where $V = \{v_1, \ldots, v_n\}$ is the set of nodes and $E = \{e_1, \ldots, e_m\}$ is the set of edges between the nodes. We use parameter ϵ_{ij} to represent the existence of an edge between vertices v_i and v_j. $\epsilon_{ij} = 1$ if there exists such an edge, $\epsilon_{ij} = 0$ otherwise. Two nodes connected by an edge are called adjacent and we denote k_i to the degree of a given vertex v_i, which is the number of edges connecting v_i with other nodes. The shortest path length between vertices v_i and v_j is denoted d_{ij}. We assume the graph G is connected and, therefore, d_{ij} is finite and positive $\forall i, j$.

Let $|\Gamma_i|$ be the number of connections between the nearest neighbors of a node $v_i \in V$ and $C_i = \frac{|\Gamma_i|}{k_i(k_i-1)}$, the *clustering coefficient* C is calculated as $C = \frac{1}{n}\sum_i C_i$. The *path length* is calculated as $L = \frac{1}{n(n-1)}\sum_{i \neq j} d_{ij}$.

Small-world networks are characterized by a high clustering coefficient and a small path length. They can be generated by randomly replacing a fraction p of the links of a d-dimensional lattice with new random links. Since the new links decrease the shortest path length between the connected nodes, they are usually called *shortcuts*. The two limiting values of $p = 0$ and $p = 1$ respectively correspond to a regular lattice and a random graph. p is commonly called the *rewiring probability*.

Different patterns can be identified in the connectivity of small-world networks resulting in a classification (Amaral et al., 2000) of these networks in scale-free networks, broad-scale networks and single-scale networks. For more details on small-world and other complex networks, (Amaral et al., 2000; Barthélémy and Amaral, 1999; Dorogovtsev et al., 2007) can be consulted.

2.2 Factor graphs

Factor graphs (Kschischang et al., 2001) are bipartite graphs with two different types of nodes: variable nodes and factor nodes. Each variable node identifies a single variable X_i that can take values from a (usually discrete) domain, while factor nodes f_j represent different functions whose arguments are subsets of variables. This is graphically represented by edges that connect a particular function node with its variable nodes (arguments).

Factor graphs are appropriate to represent those cases in which the joint probability distribution can be expressed as a factorization of several local functions:

$$p(x_1, \ldots, x_n) = \frac{1}{Z}\prod_{j \epsilon J} f_j(\boldsymbol{x_j}) \qquad (1)$$

where $Z = \sum_{\boldsymbol{x}} \prod_{j \epsilon J} f_j(\boldsymbol{x_j})$ is a normalization

constant, n is the number of variable nodes, J is a discrete index set, $\boldsymbol{X_j}$ is a subset of $\{X_1, \ldots, X_n\}$, and $f_j(\boldsymbol{x_j})$ is a function containing the variables of $\boldsymbol{X_j}$ as arguments.

The structure of a factor graph can be determined from a given undirected network by associating a factor node to each edge in the network. We use factor graphs as the base graphical for the application of LBP.

3 Loopy belief propagation

LBP works by exchanging messages between nodes. Each node sends and receives messages until a stable situation is reached. Messages, locally calculated by each node, comprise statistical information concerning neighbor nodes.

When applied on factor graphs, two kinds of messages are identified (Yedidia et al., 2005): messages $n_{i\to a}(x_i)$ sent from a variable node i to a factor node a, and messages $m_{a\to i}(x_i)$ sent from a factor node a to a variable node i.

Messages are updated according to the following rules:

$$n_{i\to a}(x_i) := \prod_{c \epsilon N(i)\backslash a} m_{c\to i}(x_i) \tag{2}$$

$$m_{a\to i}(x_i) := \sum_{\boldsymbol{x_a}\backslash x_i} f_a(\boldsymbol{x_a}) \prod_{j \epsilon N(a)\backslash i} n_{j\to a}(x_j) \tag{3}$$

$$m_{a\to i}(x_i) := \arg\max_{\boldsymbol{x_a}\backslash x_i} \{f_a(\boldsymbol{x_a}) \prod_{j \epsilon N(a)\backslash i} n_{j\to a}(x_j)\} \tag{4}$$

where $N(i)\backslash a$ represents all the neighboring factor nodes of node i excluding node a, and $\sum_{\boldsymbol{x_a}\backslash x_i}$ expresses that the sum is completed taking into account all the possible values that all variables except X_i in $\boldsymbol{X}_a$ can take –while variable X_i takes its x_i value.

Equations 2 and 3 are used when marginal probabilities are looked for (sum-product). By contrast, in order to obtain the most probable configurations (max-product), Equations 2 and 4 should be applied.

When the algorithm converges (i.e. messages do not change), marginal functions (sum-product) or max-marginals (max-product), $g_i(x_i)$, are obtained as the normalized product of all messages received by X_i:

$$g_i(x_i) \propto \prod_{a \epsilon N(i)} m_{a\to i}(x_i) \tag{5}$$

Regarding the max-product approach, when the algorithm converges to the most probable value, each variable in the optimal solution is assigned the value given by the configuration with the highest probability at each max-marginal.

4 FlexLBP program

As described in the previous sections, LBP is a widely studied and used algorithm and has been rediscovered and adapted repeatedly to particular problems. Thus, different implementations have been developed since the algorithm was first proposed, although most of them focus on a particular scheduling policy, stopping criterion, etc.

In our LBP implementation (FlexLBP), we have designed a flexible tool, so that researchers can tune different parameters according to the characteristics of the problem they are facing. The FlexLBP implementation has been done following a distributed scheme. That is, each node runs independently, triggered by the message(s) it receives. Additionally, different scheduling policies (asynchronous, synchronous, and even particular rules for individual nodes) can be selected. In addition, the set of nodes that start sending messages can be customized, the order in which messages are processed can be changed, different values can be set for the initial messages, and max-product or sum-product algorithms can be selected. Finally, and due to the distributed scheme, stopping conditions are checked locally (in each node). We have established four different stopping situations: (1) a given maximum number of iterations is reached (that is, calculated messages are different), (2) a node stops because all its neighbors have stopped, (3) message values calculated by a node have not changed in the last

i iterations, and (4) message values calculated by a node follow a periodic (cyclic) sequence.

To investigate the behavior of LBP, we collect a number of statistics that will be used in the analysis of the algorithm behavior. These statistics include information about the four different stopping criteria previously explained.

5 LBP on networks of different topologies

Key to the evaluation of LBP is the determination of the network topologies from which the factor graphs are constructed by associating a factor node to each edge in the graph. We consider two different scenarios to investigate the effects of the network topology. These scenarios are defined by the way the networks are generated.

5.1 From grids to random graphs

We start from a factor graph G^i constructed from a 2-dimensional grid with periodic boundary conditions. In G^i, there is a factor between any pair of nodes that are neighbors in the grid. The number of nodes is $n = \times m$ where m is the dimension of the grid. The number of factors is $2n$. The function values for each of the factors are independently generated.

¿From graph G^i, a collection of factor graphs is generated by rewiring the original edges in the grid of G^i with probability p. To generate a rewired graph from G^i, each edge is visited and a decision about rewiring is made with probability p. If the edge if rewired, the variable nodes in the corresponding factor node are modified but the factor node's function values are kept intact.

5.2 Adding shortcuts

Similar to the previous section, we start from a factor graph G^i constructed from a 2-dimensional grid with periodic boundary conditions. In this case, a collection of factor graphs is generated from graph G^i by adding e edges to the grid of G^i and associating a factor to each edge. Three classes of graphs are created according to the way shortcuts are added:

1. Randomly: Edges are added between two randomly selected nodes.

2. Max. distance: At each step, an edge is added between a pair of nodes at maximum distance in the current network. Every time an edge is added, distances are recalculated.

3. Min. distance: At each step, an edge is added between a pair of nodes at distance 2 in the current network. Every time an edge is added, distances are recalculated.

The different procedures used to add the edges determine differences between the path lengths of graphs belonging to the different groups. Regarding the factor nodes, no matter which method was used for adding the shortcuts, the function values for all the configurations of each of the added factors are set to 1. This means that the contribution of all possible configurations of these factors to the global function will be the same and therefore the maximum configuration of the original graph (respectively the marginals) will not be modified by the addition of the factors. However, the introduction of the new factors (or shortcuts) may have an effect on the LBP dynamics and this is precisely what we would like to identify.

6 Experiments

6.1 Design of the experiments

The starting graph structures used in our experiments are 2-dimensional grids ($m = 7$) with periodic conditions. We use binary variables ($n = 49$) and the maximum size of the factor nodes is 2. Random functions are used. The values corresponding to each factor node entry are generated as $J_{ij} = e^{\beta}$, where β is a value uniformly chosen from $(0, 1)$. To determine whether differences between the behavior of LBP on the different classes of networks are statistically significant the Kruskal-Wallis test (Hsu, 1996) has been employed.

In all the following experiments, the maximum number of messages calculated by each

node was set to 2500 and a node is said to converge when the same message value is repeated 500 times.

6.2 Investigating LBP when the rewiring probability is increased

In these experiments we investigate how LBP is affected by the changes in the rewiring probability p. Since the transition to the small-network topology is known to occur for small p values, we generate networks for values of $p \in \{0.01, 0.02, \ldots, 0.1\}$. In addition, and in order to observe the behavior of LBP when p is further increased, we also generate networks for $p \in \{0.2, 0.3, \ldots, 1.0\}$.

We identify each member of the graph collection generated from G^i with a unique assignment to parameters $p \in \{0.01, 0.02, \ldots, 0.1, 0.2, \ldots, 1.0\}$, $inst \in \{1, \ldots, 100\}$. Thus, for each value of p, 100 different graphs are generated by rewiring the edges from G^i with probability p. The total number of factor graphs generated starting from G^i is 1900. Since in our experiments we conducted experiments with 10 initial graphs, i.e. $i \in \{1, \ldots, 10\}$, and the max and sum versions of LBP were used, the total number of LBP runs was $38,000$.

Figures 1, 2 and 3 show the results of max-LBP for different values of the rewiring probability. For each of the descriptors employed, the results were computed as the average among all the runs for the 10 different instances. The used descriptors were: the function value of the best solution found by LPB, the number of nodes that converged and the number of iterations for nodes that converged.

An analysis of Figure 1 reveals the decrease in the average value of the best solution found by max-LBP when the rewiring probability is increased. This may indicate that the type of constraints introduced by higher values of p determine smaller values of the optimum and/or that it is more difficult for max-LBP to find the actual optimum of the functions. Evidence of the difficulties of max-LBP for converging when p is increased emerges from the analysis of Figures 2 and 3. It can be seen that the number of

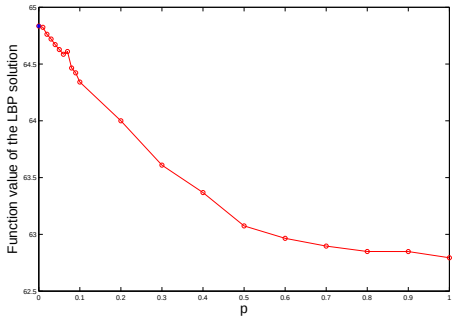


Figure 1: Value of the best solution found by max-LBP for different values of the rewiring probabilities.

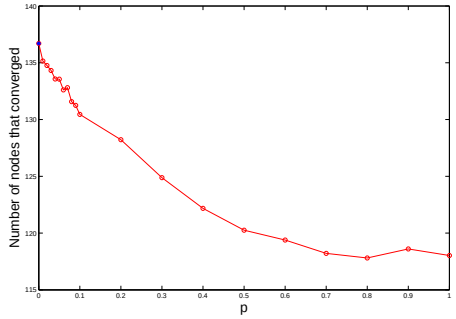


Figure 2: Number of nodes that converged when using max-LBP with different values of the rewiring probabilities.

nodes that converged decreases with p. On the other hand, the number of iterations needed by the nodes that converged is higher. Although the curve describing the number of nodes that converged is clearly monotonically decreasing, it seems to be more pronounced as p goes from 0.1 to 0.5 than for $p > 0.5$. This fact indicates that the behavior of max-LBP is more sensitive to changes around certain values or p.

We conducted similar experiments using sum-LBP for the same set of graphs. As it has been previously reported (Mooij et al., 2007), LBP convergence is easier to achieve for the sum case than for the max case. Figures 4 and 5 respectively show the number of nodes that converge and the average number of iterations for different values of the rewiring probability. Al-

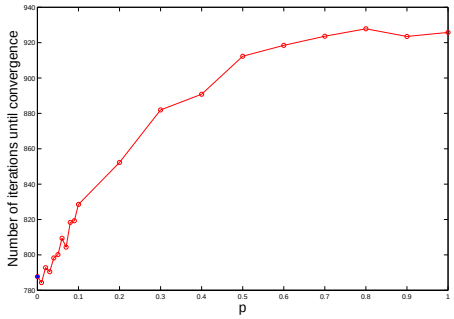


Figure 3: Number of iterations for nodes that converged when using max-LBP with different values of the rewiring probabilities.

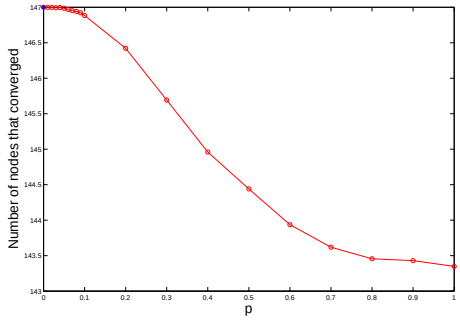


Figure 4: Number of nodes that converged when using sum-LBP with different values of the rewiring probabilities.

though the number of nodes that converge is higher, the curve is also monotonically decreasing. Conversely, the number of iterations until convergence is increased. Therefore, the influence of the rewiring probability seems to be the same for max-LBP and sum-LBP.

Results shown in the previous figures correspond to the average for 10 different network topologies.

For each descriptor we have carried out the Kruskal-Wallis statistical test to compare the LBP results for each possible pair of values of p. For some pairs of values and descriptors, the test did not find statistically significant differences.

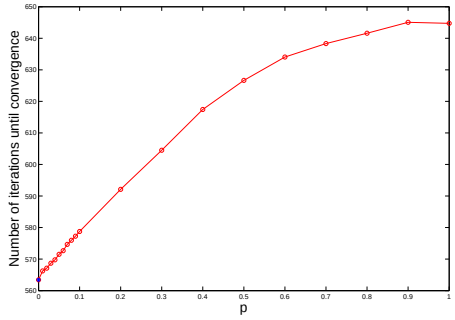


Figure 5: Number of iterations for nodes that converged when using sum-LBP for different values of the rewiring probabilities.

6.3 Investigating the influence of factor additions

The purpose of the following experiments is twofold: First, we investigate the way in which the addition of shortcuts can modify the behavior of LBP. Second, we analyze the influence that the way in which shortcuts are added has in the LBP behavior. We are particularly interested in knowing whether the addition of shortcuts leads to improvements in the results achieved by LBP, i.e. better solutions are obtained or the algorithm converges faster.

The number of added edges was fixed to $e = 10$ and the number of initial graphs to 50. An instance corresponds to a random assignment of the function values to the factor nodes defined on the 2-dimensional grid.

For each of the three methods used to add the edges described in Section 5.2, 50 different graph structures are created by considering 50 possible ways of adding 10 edges to the original grid. The total combination of the initial instances, the methods used to add the edges, and the graph structures generated for each method was $50 \times 3 \times 50 = 7500$. max-LBP and sum-LBP were run on each of these instances.

For each of the instances, we compute the average value of the best solutions found by max-LBP among the 50 graph structures for each of the three methods employed to add the shortcuts. Average values are then compared with the value found by max-LBP in the original graph (without added shortcuts). Of the 50

graphs, the Random, Max. distance and Min. distance methods respectively improve (on average) the values of the function for 9, 9, and 10 of the instances. They respectively converged to the same optimal solution for 22, 22, and 24 instances. These results show that adding shortcuts may have an effect on the quality of the solution obtained, at least in some cases.

To find statistical differences between the behavior of the three methods, the Kruskal-Wallis test was applied using the values of the best solution found. It found statistical significant differences between at least two of the three methods in 13 of the instances. The average gain in the value of the function is shown in Figure 6a. Negative values indicate that the solution found by LBP in the graphs with shortcuts was worse than in the original graph. Of the 13 instances, the method that reduces the path length the most (i.e. Min. distance) was the best in improving the value with respect to the other two methods in 9 of the instances.

On the 37 instances in which the statistical test did not find significant differences between any pair of the methods regarding the quality of the solution obtained, we conducted additional tests to identify significant differences in the number of (factor and vertices) nodes that converged and the number of iterations to convergence. In 21 of the instances, significant differences were found in the number of nodes that converged. The gain in the proportion of nodes that converged with respect to the initial graph is shown in Figure 6b. The Min. distance method achieved a higher proportion of nodes that converged in 18 of the 21 instances. Finally, regarding the number of iterations until convergence, there were significant differences in 20 of the 37 instances and, as shown in Figure 6c), in 12 of them Min. distance achieved a higher number of iterations.

In general, it was observed that max-LBP obtained better solutions and converged in a higher proportion of the nodes of the graphs generated by the Min. distance method than the other two methods. However, the average number of iterations also increased.

Concerning the behavior of sum-LBP, there were not significant differences in the number of nodes that converged. Nevertheless, a different trend to that shown for max-LBP was manifested. As observed in Figure 7, the Min. distance method needed fewer iterations than the other two for all of the instances.

7 Conclusions and future work

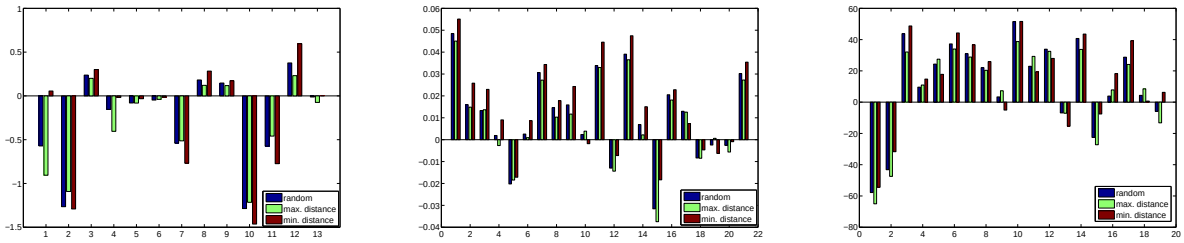
In this paper we have analyzed the effect of the network topology on the behavior of LBP. The empirical analysis of the LBP results for the different graphs considered has shown that finding optimal solutions is harder for LBP when the rewiring probability is increased.

On the other hand, we have shown that adding shortcuts to the initial lattice graphs changes the dynamics of LBP in a less clear way. These changes may determine that better solutions are achieved and/or the number of iterations to convergence can be diminished. The results obtained seem to indicate that the method which reduces the path length the most can lead to more improvements in the LBP behavior than selecting the shortcuts randomly, or choosing them in such a way that the path length is minimally reduced.

Although we have not advanced a method or heuristic for choosing the right shortcuts, i.e. those that can help LBP to escape from cycles or converge to better solutions, we speculate that an influencing factor to this respect is the reduction in the local and global distances determined by the addition of the shortcut. Furthermore, we have just investigated the addition of edges in a step previous to the algorithm start. It might be the case that the adequacy of adding a particular shortcut will change dynamically according to the current step of the LBP algorithm. An open question is then to devise ways to add shortcuts *on line* taking into account the current state of the process.

Acknowledgements

This work has been partially supported by the Etortek, Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2005-03824 and Consolider Ingenio 2010 - CSD2007-00018 projects (Span-



a) b) c)

Figure 6: Influence of the network topology in the behavior of max-LBP when shortcuts are added to an initial grid configuration. a) Improvements in the function values of the best solution found by LPB. b) Number of nodes that converged. c) Number of iterations for nodes that converged.

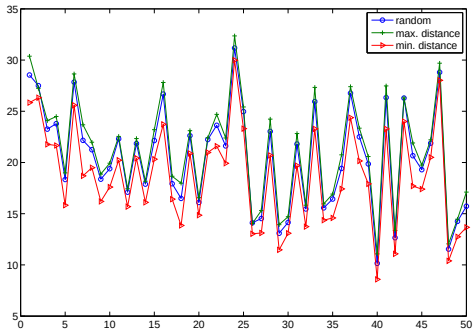


Figure 7: Influence of the network topology in the number of iterations before convergence of sum-LBP when shortcuts are added.

ish Ministry of Education and Science) and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

References

L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. 2000. Classes of small-world networks. *Proceedings of the National Academy of Sciences (PNAS)*, 97(21):11149–11152.

M. Barthélémy and L. A. N. Amaral. 1999. Small-world networks: Evidence for a crossover picture. *Physical Review Letters*, 82(15):3180–3183.

S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. 2007. Critical phenomena in complex networks. *arXiv.org*, arXiv:0705.0020v6 [cond-mat.stat-mech], Nov.

G. Elidan, I. McGraw, and D. Koller. 2006. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the Twenty-second Annual Conference on Uncertainty in Artificial Intelligence (UAI-2006)*, Boston, Massachussetts.

R. Ferrer i Cancho, C. Janssen, and R. V. Solé. 2001. Topology of technology graphs: Small world patterns in electronic circuits. *Physical Review E. Statistical, Nonlinear, and Soft Matter Physics*, 64:046119.

J. C. Hsu. 1996. *Multiple Comparisons: Theory and Methods*. Chapman and Hall.

F. R. Kschischang, B. J. Frey, and H. A. Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

J. M. Mooij, B. Wemmenhove, H. J. Kappen, and T. Rizzo. 2007. Loop corrected belief propagation. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS-07)*.

J. Ohkubo, M. Yasuda, and K. Tanaka. 2005. Statistical-mechanical iterative algorithms on complex networks. *Physical Review E*, 72(046135):8 pages.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California.

D.J. Watts and S.H. Strogatz. 1998. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. 2005. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312.

Factorized Normalized Maximum Likelihood Criterion for Learning Bayesian Network Structures

Tomi Silander, Teemu Roos, Petri Kontkanen and Petri Myllymäki
Helsinki Institute for Information Technology HIIT, Finland

Abstract

This paper introduces a new scoring criterion, factorized normalized maximum likelihood, for learning Bayesian network structures. The proposed scoring criterion requires no parameter tuning, and it is decomposable and asymptotically consistent. We compare the new scoring criterion to other scoring criteria and describe its practical implementation. Empirical tests confirm its good performance.

1 Introduction

The popular Bayesian criterion, BDeu (Buntine, 1991), for learning Bayesian network structures has recently been reported to be very sensitive to the choice of prior hyperparameters (Silander et al., 2007). On the other hand, general model selection criteria, such as AIC (Akaike, 1973) and BIC (Schwarz, 1978), are derived through asymptotics and their behavior is suboptimal for small sample sizes. The study of different scoring criteria is further complicated by the fact that learning the network structure is NP-hard for all popular scoring criteria (Chickering, 1996), even if these criteria have a convenient characteristic of decomposability, which allows incremental scoring in heuristic local search (Heckerman et al., 1995). Due to recent advances in exact structure learning (Koivisto and Sood, 2004; Silander and Myllymäki, 2006) it is feasible to find the optimal network for decomposable scores when the number of variables is less than about 30. This makes it possible to study the behavior of different scoring criteria without the uncertainty stemming from the heuristic search.

In this paper we introduce a new decomposable scoring criterion for learning Bayesian network structures, the *factorized normalized maximum likelihood* (fNML). This score features no tunable parameters, thus avoiding the sensitivity problems of Bayesian scores. We show that the new criterion is asymptotically consistent. Unlike AIC and BIC, it is derived based on optimality criterion for finite sample sizes, and it has a probabilistic interpretation.

The rest of the paper is structured as follows. In Section 2, we will first introduce Bayesian networks and the notation needed later. In Section 3, we review the most popular decomposable scores, after which in Section 4, we are ready to introduce the fNML criterion. We then briefly discuss the implementation of this new score in Section 5. Section 6 presents the empirical experiments, and the conclusions are summarized in Section 7.

2 Bayesian Networks

We assume that reader is familiar with Bayesian networks (for tutorial, see (Heckerman, 1996)), and only introduce the notation needed later in this paper.

A Bayesian network defines a joint probability distribution for an m-dimensional multivariate data vector $X = (X_1, \ldots, X_m)$. We assume that all variables are discrete, so that variable X_i may have r_i different values $\{1, \ldots, r_i\}$.

A Bayesian network consists of a directed acyclic graph G and a set of conditional probability distributions. We specify the DAG with a vector $G = (G_1, \ldots, G_m)$ of parent sets so that $G_i \subset \{X_1, \ldots, X_m\}$ denotes the parents of variable X_i, i.e., the variables from which there is an arc to X_i. Each parent set G_i has q_i ($q_i = \prod_{X_p \in G_i} r_p$) possible values that are the

possible value combinations of the variables belonging to G_i. We assume a non-ambiguous enumeration of these values and denote the fact that G_i holds the j^{th} value combination simply by $G_i = j$.

The local Markov property for Bayesian networks states that each variable is independent of its non-descendants given its parents. Functionally this is equivalent to the following factorization of the joint distribution

$$P(x \mid G) = \prod_{i=1}^{m} P(x_i \mid G_i). \quad (1)$$

The conditional probability distributions $P(X_i \mid G_i)$ are determined by a set of parameters, Θ, via the equation

$$P(X_i = k \mid G_i = j, \Theta) = \theta_{ijk},$$

where k is a value of X_i, and j is a value configuration of the parent set G_i. We denote the set of parameters associated with variable X_i by Θ_i.

For learning Bayesian network structures we assume a data D of N complete i.i.d instantiations of the vector X, i.e., an $N \times m$ data matrix without missing values. It turns out to be useful to introduce a notation for certain parts of this data matrix. We often want to select rows of the data matrix by certain criteria. We then write the selection criterion as a superscript of the data matrix D. For example, $D^{G_i=j}$ denotes those rows of D where the variables of G_i have the j^{th} value combination. If we further want to select certain columns of these rows, we denote the columns by subscripting D with a corresponding variable set. As a shorthand, we write $D_{\{X_i\}} = D_i$. For example, $D_i^{G_i=j}$ selects the i^{th} column of the rows $D^{G_i=j}$.

Since the rows of D are assumed to be i.i.d, the probability of a data matrix can be calculated just by taking the product of the row probabilities. Combining equal terms yields

$$P(D \mid G, \Theta) = \prod_{i=1}^{m} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}}, \quad (2)$$

where N_{ijk} denotes number of rows in $D^{X_i=k, G_i=j}$.

For a given structure G, we use notation $\hat{P}(D \mid G) = \sup_\theta P(D \mid G, \theta)$. The maximizing parameters are simply the relative frequencies found in data: $\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$, where N_{ij} denotes the number of rows in $D^{G_i=j}$, or 1.0 if $N_{ij} = 0$. We often drop the dependency on G when it is clear from the context.

3 Decomposable scores

In general, a scoring function $\textsc{Score}(G, D)$ for learning a Bayesian network structure is called decomposable, if it can be expressed as a sum of local scores

$$\textsc{Score}(G, D) = \sum_{i=1}^{m} S(D_i, D_{G_i}). \quad (3)$$

Many popular scoring functions avoid overfitting by balancing the fit to the data with the complexity of the model. A common form of this idea can be expressed as

$$\textsc{Score}(G, D) = \log \hat{P}(D \mid G) - \Delta(D, G), \quad (4)$$

where $\Delta(D, G)$ is a complexity penalty.

The maximized likelihood $\hat{P}(D \mid G)$ decomposes by the network structure, and for the decomposable scores handled in this paper, the complexity penalty decomposes too. Hence, we can write the penalized scores in the decomposed form (3), with the local scores given by

$$S(D_i, D_{G_i}) = \log \hat{P}(D_i \mid D_{G_i}) + \Delta_i(D_i, D_{G_i}). \quad (5)$$

Different scores differ in how the local penalty $\Delta_i(D_i, D_{G_i})$ is determined.

3.1 AIC and BIC

Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are two popular decomposable scores for learning Bayesian network structures. These scores do not have any additional parameters so in this sense they are similar to the proposed fNML score. The penalty terms for these scores are $\Delta_i^{\text{BIC}} = \frac{q_i(r_i-1)}{2} \ln N$, and $\Delta_i^{\text{AIC}} = q_i(r_i - 1)$. Both of these complexities are independent of the data, and only depend on the arities r_i of random variables and the structure of the Bayesian network.

3.2 Bayesian Dirichlet scores

Bayesian Dirichlet (BD) scores assume that the parameter vectors Θ_{ij} are independent of each other and distributed by Dirichlet distributions with hyper-parameter vector $\vec{\alpha}_{ij}$. Given a vector of hyper-parameters $\vec{\alpha}$, the local score can be written as

$$\begin{aligned} S_{\mathrm{BD}}(D_i, D_{G_i}, \vec{\alpha}) &= \log P(D_i \mid D_{G_i}, \vec{\alpha}) \\ &= \sum_{j=1}^{q_i} \log P(D_i^{G_i=j} \mid D_{G_i}^{G_i=j}, \vec{\alpha}_{ij}) \\ &= \sum_{j=1}^{q_i} \log \left(\frac{B(\vec{\alpha}_{ij} + \vec{N}_{ij})}{B(\vec{\alpha}_{ij})} \right), \end{aligned}$$

where B is a multinomial Beta function

$$B(\alpha_1, \ldots, \alpha_K) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_{i=1}^{K} \alpha_k)}.$$

With all $\alpha_{ijk} = 1$ we get a K2-score (Cooper and Herskovits, 1992), and with $\alpha_{ijk} = \frac{\alpha}{q_i r_i}$ we get a family of BDeu scores popular for giving equal scores to different Bayesian network structures that encode the same independence assumptions. BDeu scores depend only on a single parameter, the *equivalent sample size* α. Recent studies on the role of this parameter show that network learning under BDeu is very sensitive to this parameter (Silander et al., 2007).

For comparison, we can write the BD-score as a penalized maximized likelihood with penalty

$$\Delta_i^{BD}(D_i, D_{G_i}) = \sum_{i=i}^{q_i} \log \left(\frac{\hat{P}(D_i^{G_i=j} \mid D_{G_i}^{G_i=j})}{P(D_i^{G_i=j} \mid D_{G_i}^{G_i=j}, \vec{\alpha}_{ij})} \right). \quad (6)$$

We immediately notice that this penalty is always positive. The complexity is data-dependent and it is controlled by the hyper-parameters α_{ijk}. The asymptotic behavior of this Bayesian regret is well studied (Grünwald, 2007). However, when learning Bayesian networks, the data parts $D_i^{G_i=j}$ are often very small, which makes asymptotic result less informative.

4 fNML

The *factorized normalized maximum likelihood* (fNML) score is based on the *normalized maximum likelihood* (NML) distribution (Shtarkov, 1987; Rissanen, 1996). The NML distribution for the model class $\mathcal{M}$ (which may or may not be a Bayesian network) is the unique distribution solving the minimax problem

$$\min_{Q} \max_{D'} \frac{\hat{P}(D' \mid \mathcal{M})}{Q(D' \mid \mathcal{M})}, \quad (7)$$

where Q ranges over all distributions.

As originally shown by Shtarkov (1987) the solution of the above minimax problem is given by

$$P_{\mathrm{NML}}(D \mid \mathcal{M}) = \frac{\hat{P}(D \mid \mathcal{M})}{\sum_{D'} \hat{P}(D' \mid \mathcal{M})}, \quad (8)$$

where the normalization is over all data sets D' of a fixed size N. The log of the normalizing factor is called *parametric complexity* or *regret*.

Evaluation of the normalizing sum is often hard due to exponential number of terms in the sum. Currently, there are tractable formulas for only a handful of models; for examples, see (Grünwald, 2007). In the case of a single r-ary multinomial variable and the sample size n the normalizing sum is given by

$$\mathcal{C}_n^r = \sum_{k_1+k_2+\ldots+k_r=n} \frac{n!}{k_1!\, k_2! \cdots k_r!} \prod_{j=1}^{r} \left(\frac{k_j}{n} \right)^{k_j}, \quad (9)$$

where the sum goes over all non-negative integer vectors $(k_j)_{j=1}^r$ that sum to n. A linear-time algorithm for the computation of $\mathcal{C}_n^r$ was introduced recently by Kontkanen and Myllymäki (2007).

Given a data set D, the NML model selection criterion proposes to choose the model $\mathcal{M}$ for which the $P_{\mathrm{NML}}(D \mid \mathcal{M})$ is largest. After taking the logarithm the score is in a form of penalized log likelihood with complexity penalty describing how well the model can fit any equal size dataset D'.

Because of the score equivalence of the maximum likelihood score, the NML score is score

equivalent as well. However, it is not decomposable, and the parent assignment problem is known to be NP-hard (Koivisto, 2006). Sacrificing the score equivalence we propose a decomposable version of this score, which penalizes the complexity locally similarly to the other decomposable scores. Specifically, we propose the local score

$$S_{\text{fNML}}(D_i, D_{G_i}) = \log P_{\text{NML}}(D_i \mid D_{G_i}) \quad (10)$$
$$= \log \left(\frac{\hat{P}(D_i \mid D_{G_i})}{\sum_{D_i'} \hat{P}(D_i' \mid D_{G_i})} \right),$$

where the normalizing sum goes over all the possible D_i-column vectors of length N, i.e., $D_i' \in \{1, \ldots, r_i\}^N$.

Since equation (10) defines a (log) conditional distribution for the data column D_i, adding these local scores together yields a total score that defines a distribution for the whole data. In this sense fNML can be seen as an alternative way to define the marginal likelihood for the data

$$\log P_{\text{fNML}}(D \mid G) = \sum_{i=1}^{m} \log P_{\text{NML}}(D_i \mid D_{G_i}).$$

At the same time, combining the local scores yields an enumerator that equals the decomposition of the maximum likelihood, thus the whole score can be seen as a penalized maximum log-likelihood with local (data-dependent) penalties

$$\Delta_i^{\text{fNML}}(D_{G_i}) = \log \sum_{D_i'} \hat{P}(D_i' \mid D_{G_i}). \quad (11)$$

The following observation follows from the factorization of the maximum likelihood by the parent configurations, and it is crucial for efficient calculation of the local penalty term.

Theorem 1. *The local penalty of fNML can be expressed in terms of multinomial normalizing constants*

$$\Delta_i^{\text{fNML}}(D_{G_i}) = \sum_{j=1}^{q_i} \log \mathcal{C}_{N_{ij}}^{r_i},$$

where $\mathcal{C}_{N_{ij}}^{r_i}$ is the normalizing constant of NML for an r_i-ary multinomial model with sample size N_{ij}.

The theorem follows by noting that the maximized likelihood $\hat{P}(D_i \mid D_{G_i})$ factorizes into independent parts according to the values of D_{G_i}.

To conclude this section we show that asymptotically, and under mild regularity conditions, the fNML score belongs to the (large) class of BIC-like scores that are consistent. Other scores in this class include most Bayesian and MDL criteria. The regularity conditions required for BIC-like behavior typically exempt a measure zero set of generating parameters, such as the boundaries of the parameter simplex. The following theorem gives sufficient conditions on the penalty term that guarantee consistency for exponential family models.

Theorem 2 (Remark 1.2 in (Haughton, 1988))**.** *For (curved) exponential families, if data is generated by an i.i.d. distribution p, and the penalty term is given by $\frac{1}{2} k \, a_N$, where k is the number of parameters and a_N is a sequence of positive real numbers, satisfying*

$$a_N / N \to 0, \quad \textit{and} \quad a_N \to \infty,$$

as $N \to \infty$, then, symptotically, the model containing p that has the least number of parameters will be chosen.

Since Bayesian networks are curved exponential families (Geiger et al., 2001; Chickering, 2002), it now remains to prove that the penalty term of fNML satisfies this property.

Theorem 3 (Asymptotically fNML behaves like BIC)**.** *Assuming that the maximum likelihood parameters are asymptotically bounded away from the boundaries of the parameter simplex, the local penalty of fNML behaves as*

$$\Delta_i^{\text{fNML}}(D_{G_i}) = \frac{q_i(r_i - 1)}{2} \log N + \mathcal{O}(1),$$

almost surely, where the $\mathcal{O}(1)$ term is bounded by a constant wrt. N.

Proof. By Thm. 1, the local penalty is a sum of logarithms of multinomial normalizing constants. The latter is known to grow as $\log \mathcal{C}_{N_{ij}}^{r_i} = \frac{r_i - 1}{2} \log N_{ij} + \mathcal{O}(1)$, (Rissanen, 1996). Under the assumption that the maximum likelihood parameters are bounded away from the

boundaries, the counts N_{ij} grow linearly in the total sample size N almost surely, which implies that we have $\log N_{ij} = \log([\eta + o(1)]N) = \log N + \mathcal{O}(1)$ with some $0 < \eta < 1$. Adding together the q_i terms yields the result. □

Since $q_i(r_i - 1)$ is the number of parameters (associated with the ith variable), the property of Thm. 2 holds for the fNML penalty.

5 Implementation

We now provide information for practical implementation of the fNML score for Bayesian networks. Due to the decomposability of the score the only new implementational issue for the fNML is to calculate the terms $\mathcal{C}_n^r$ of the Thm. 1. For reasonable N and R ($R = \max r_i$) these values can be stored in an $N \times R$ table, which can be done before structure learning. Moreover, this table does not depend on data or any parameters, so it can be done just once.

The calculation of the $\mathcal{C}$-table with N rows and R columns proceeds as follows. First of all, $\mathcal{C}_0^r = 1$ for all r, and $\mathcal{C}_n^1 = 1$ for all n. For $r = 2$ we can use the formula (9), which yields

$$\mathcal{C}_n^2 = \sum_{h=0}^{n} \binom{n}{h} \left(\frac{h}{n}\right)^h \left(\frac{n-h}{N}\right)^{n-h}, \qquad (12)$$

and for $r > 2$ we can use the recursion (Kontkanen and Myllymäki, 2007)

$$\mathcal{C}_n^r = \mathcal{C}_n^{r-1} + \frac{n}{r-2}\mathcal{C}_n^{r-2}. \qquad (13)$$

Calculating the column $\mathcal{C}_*^2$ using the formula (12) takes time $\mathcal{O}(N^2)$, and the calculation of the rest of the table using the formula (13) takes just $\mathcal{O}(NK)$. For very large N, the complexity of calculating the column $\mathcal{C}_*^2$ may be prohibitive. In this case a very accurate Szpankowski approximation (Kontkanen et al., 2003)

$$\mathcal{C}_n^2 = \frac{n\pi}{2} e^{\sqrt{\frac{8}{9n\pi}} + \frac{3\pi - 16}{36n\pi}} \qquad (14)$$

can be used.

If the space for storing the table is critical, one may just store 1000 first entries of column $\mathcal{C}_*^2$, use Szpankowski approximation for the rest of the column, and use formula (13) for calculating the values for $r > 2$.

6 Experiments

It is not obvious how to compare different criteria for learning Bayesian network structures. If the data is generated from a Bayesian network, one might call for selecting the data generating network, but if the generating network is complex, and the sample size is small, it may be rational to pick a simpler model.

This simplicity requirement is often backed up by arguments about the generalization capability of the model. However, it is not always clear how the network structure should be used for prediction.

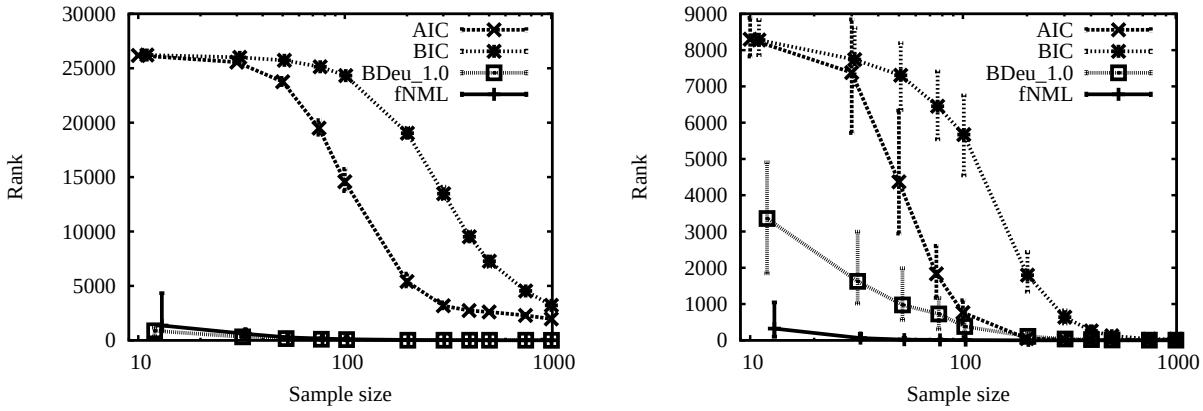
A softer version of discovering the generating model is to compute a structural distance measure between the selected and the generating network structures. A common choice is to calculate an editing distance with operations such as arc additions, deletions and reversals. Even if we take the generating structure as a golden standard, this approach is problematic, since these editing operations are not independent. For example, fixing a certain arc can lead to several other changes to the network structure if the selection by a score is made only among the structures having the fixed arc present.

Despite of these problems in the empirical testing, we conducted a golden standard experiment. We first generated data from different networks with five nodes, and then studied how the generating network structures were ranked among all the possible networks by different scoring criteria.

For BDeu and fNML scores that both calculate the probability $P(D \mid G)$, we also compared the scores for the real data sets. This experiment can be seen as the result of a sequential prediction competition, since by the chain rule we can write

$$P(D \mid G) = \prod_{i=1}^{N} P(d_i \mid G, d^{i-1}), \qquad (15)$$

where d_i is the ith data vector, and $d^{i-1} = \{d_1, \ldots, d_{i-1}\}$ denotes the first $i-1$ vectors. The idea follows the principle of prequential model selection (Dawid, 1984).



(a) BDeu Scheme

(b) Dir(1/2,...,1/2) Scheme

Figure 1: The median curves for different scoring criteria as a function of sample size when the parameters for a 5-node, 7-edge network were generated by the BDeu and Dir(1/2,...,1/2) schemes. Errorbars indicate upper and lower quartiles.

We will now explain the experiments in more detail.

6.1 Artificial data

We first compared the ability of different scoring criteria to discover the data generating structure. For this purpose we generated 100 different 5-node Bayesian network structures with 4 edges and another 100 structures with 7 edges. The variables were randomly assigned to have 2 – 4 values ($r_i \in \{2,3,4\}$). For each network, we generated parameters by two different schemes. The first scheme exactly matched the assumptions of the BDeu score with $\alpha = 1$, i.e., the parameters were distributed by $\theta_{ij} \sim Dir(\frac{1}{r_i q_j}, \ldots, \frac{1}{r_i q_j})$. The other scheme was to generate the parameters independently from a Dirichlet distribution $\theta_{ij} \sim Dir(1/2, \ldots, 1/2)$. This distribution was selected instead of the uniform distribution in order to make the generating structure more identifiable.

For each network (structure + parameters), we generated 100 data sets of 1000 data vectors, and studied how different scoring criteria ranked the structure of the generating network among all the 5-node networks as a function of (sub)sample size.

Not surprisingly, the results indicate that when parameter generation mechanism matches the assumptions of the BDeu-score, the BDeu usually also ranks the generating structure higher than the other scores (Figure 1(a)). However, fNML usually behaves very similarly to BDeu. The density of the network (4 vs. 7 edges) is not a very significant factor. If anything, the similar behavior of fNML and BDeu is more pronounced in networks with 7 edges. For the parameter-free scores, AIC and BIC, the underfitting tendency of BIC can be clearly detected whereas AIC tends to rank the generating network higher. Qualitatively these two scores seem to behave similarly to each other.

Switching the parameter generation scheme to independent Dirichlets with $\alpha_{ijk} = 0.5$ usually also switches the ranking ability of fNML and BDeu, while the behavior of AIC and BIC stays mostly unaffected. For example, Figure 1(b) was generated using the same network structure as for Figure 1(a). Only the parameter generation scheme was changed from BDeu to Dir. For dense networks fNML often appears as a clear winner.

6.2 Real data

Learning the structure with AIC or BIC does not readily suggest any particular way to use the learned structure for prediction, but the prequential interpretation of the BDeu score and the fNML allows comparison. However, the BDeu score is known to be very sensitive to the

Table 1: Summary of the prediction experiment.

Data	N	m	#vals	α^*	BDeu1	BDeu*	fNML
balance	625	5	4.6	48	-4549.06	**-4445.64**	-4478.36
iris	150	5	3.0	2	-452.21	**-449.71**	-450.90
thyroid	215	6	3.0	2	-577.52	-575.55	**-572.42**
liver	345	7	2.9	4	-1309.67	-1299.83	**-1299.38**
ecoli	336	8	3.4	8	-1715.92	-1661.34	**-1643.64**
abalone	4177	9	3.0	6	-15946.58	-15891.25	**-15847.33**
diabetes	768	9	2.9	4	-3678.57	-3662.31	**-3654.02**
post operative	90	9	2.9	3	-647.35	-642.98	**-639.94**
yeast	1484	9	3.7	6	-7938.60	-7873.21	**-7848.98**
breast cancer	286	10	4.3	8	-2781.62	**-2737.20**	-2739.34
shuttle	58000	10	3.0	3	*-97635.72*	**-97620.78**	-97714.22
tic tac toe	958	10	2.9	51	-9423.07	**-9126.78**	-9162.39
bc wisconsin	699	11	2.9	8	-3315.51	-3262.33	**-3239.56**
glass	214	11	3.3	6	-1288.93	-1255.73	**-1233.18**
page blocks	5473	11	3.2	3	-12455.60	-12438.01	**-12410.69**
heart cleveland	303	14	3.1	13	-3450.07	-3356.78	**-3352.32**
heart hungarian	294	14	2.6	5	-2376.53	-2348.23	**-2343.65**
heart statlog	270	14	2.9	10	-2867.54	-2819.37	**-2814.28**
wine	178	14	3.0	8	-1866.41	-1821.28	**-1808.66**
adult	32561	15	7.9	50	-329373.73	-326803.91	**-326486.85**

equivalent sample size parameter, which creates an extra complication.

For predictive comparison we selected 20 UCI data sets[1] for which the score maximizing hyperparameter α has been reported (Silander et al., 2007), and we compared the maximum fNML scores to the maximum scores obtained with BDeu1 (BDeu with $\alpha = 1.0$) and BDeu* (BDeu with score maximizing α). In reality, we do not know the score maximizing α's, and searching structures with many α is usually computationally too hard. Optimal structures were obtained by the exact structure learning algorithm described in (Silander and Myllymäki, 2006).

Table 1 lists for each data set the number of data vectors N, the number of variables m, the average number of values per variable #vals, the BDeu maximizing equivalent sample size parameter $\alpha*$ (with integer precision), and the actual scores obtained with three different scoring criteria. The score obtained with fNML is the best of the three 14 times out of 20, and only once BDeu1 yields higher score than fNML.

[1] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

7 Conclusions

We have introduced a new probabilistic scoring criterion, the factorized normalized maximum likelihood, for learning Bayesian network structures from complete discrete data. The score aims at being an efficient and parameter-free criterion for finite sample sizes. The score is also decomposable, which makes it possible to use it with existing search heuristics and exact structure learning algorithms.

Initial empirical tests are promising. We are particularly pleased with fNML's ability to learn network structures with good predictive capabilities. While lot more empirical work has to be done, the current experiments already show a great promise for a good and care free

scoring criterion for learning Bayesian network structures.

Acknowledgments

The authors thank the anonymous referees for valuable comments. This work was supported in part by the Academy of Finland (Project Civi), by the Finnish Funding Agency for Technology and Innovation (Kukot, PMMA), and the IST Programme of the European Community, under the PASCAL Network of Excellence.

References

H. Akaike. 1973. Information theory and an extension of the maximum likelihood principle. In B.N. Petrox and F. Caski, editors, *Proceedings of the Second International Symposium on Information Theory*, pages 267–281, Budapest. Akademiai Kiado.

W. Buntine. 1991. Theory refinement on Bayesian networks. In B. D'Ambrosio, P. Smets, and P. Bonissone, editors, *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann Publishers.

D.M. Chickering. 1996. Learning Bayesian networks is NP-Complete. In D. Fisher and H. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag.

D.M. Chickering. 2002. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

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

A.P. Dawid. 1984. Statistical theory: The prequential approach. *Journal of the Royal Statistical Society A*, 147:278–292.

D. Geiger, D. Heckerman, H. King, and C. Meek. 2001. Stratified exponential families: graphical models and model selection. *Annals of Statistics*, 29:505–529.

P. Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press.

D.M.A. Haughton. 1988. On the choice of a model to fit data from an exponential family. *Annals of Statistics*, 16(1):342–355.

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, September.

D. Heckerman. 1996. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division, One Microsoft Way, Redmond, WA 98052.

M. Koivisto and K. Sood. 2004. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, May.

M. Koivisto. 2006. Parent assignment is hard for the MDL, AIC, and NML costs. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT-06)*, pages 289–303.

P. Kontkanen and P. Myllymäki. 2007. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6):227–233.

P. Kontkanen, W. Buntine, P. Myllymäki, J. Rissanen, and H. Tirri. 2003. Efficient computation of stochastic complexity. In C. Bishop and B. Frey, editors, *Proceedings of the Ninth International Conference on Artificial Intelligence and Statistics*, pages 233–238. Society for Artificial Intelligence and Statistics.

J. Rissanen. 1996. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January.

G. Schwarz. 1978. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464.

Yu.M. Shtarkov. 1987. Universal sequential coding of single messages. *Problems of Information Transmission*, 23:3–17.

T. Silander and P. Myllymäki. 2006. A simple approach for finding the globally optimal Bayesian network structure. In R. Dechter and T. Richardson, editors, *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 445–452. AUAI Press.

T. Silander, P. Kontkanen, and P. Myllymäki. 2007. On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. In R. Parr and L. van der Gaag, editors, *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 360–367. AUAI Press.

Large Incomplete Sample Robustness in Bayesian Networks

Jim Q. Smith
Department of Statistics
University of Warwick
Coventry, CV4 7AL, UK

Alireza Daneshkhah
Department of Management Science
University of Strathclyde
Glasgow, G1 1QE, UK

Abstract

Under local DeRobertis (LDR) separation measures, the posterior distances between two densities is the same as between the prior densities. Like Kullback - Leibler separation they also are additive under factorization. These two properties allow us to prove that the precise specification of the prior will not be critical with respect to the variation distance on the posteriors under the following conditions. The genuine and approximating prior need to be similarly rough, the approximating prior has concentrated on a small ball on the margin of interest, not on the boundary of the probability space, and the approximating prior has similar or fatter tails to the genuine prior. Robustness then follows for all likelihoods, even ones that are misspecified. Furthermore, the variation distances can be bounded explicitly by an easy to calculate function of the prior LDR separation measures and simple summary statistics of the functioning posterior. In this paper we apply these results to study the robustness of prior specification to learning Bayesian Networks.

1 Introduction

Discrete Bayesian networks (BNs) are now widely used as a framework for inference. The usual Bayesian methodology requires the selection of prior distributions on the space of conditional probabilities and various authors have suggested ways to do this (see (Cowell et al, 2000) and references therein). When data sets are complete, the usual analysis is conjugate and it is straightforward to appreciate the effect of prior specification on the subsequent inferences. However it is now more common to be working on problems where data entries are randomly or systematically missing. In this case conjugacy is then lost, models can become unidentifiable and sensitive to outliers. In such circumstances it is much less clear what features of the prior drive the inferential conclusions. Of course good modelers use various forms of sensitivity analyses to examine potential prior influence. However it is hard to do this systematically and to be sure that the posterior densities used really are robust to prior specifications, even when the sample size n is large. Indeed results on local sensitivity in Gustafson and Wasserman (1995) appeared to suggest that the hoped for robustness is a vain one.

A new family of separation measures has now been discovered which encode neighbourhoods of a prior that are on the one hand plausibly large and on the other are sufficient to enable the modeler to determine posterior variation neighbourhoods within which all posterior densities arising from the prior neighbourhood must lie. These posterior total variation neighbourhoods can be bounded explicitly in terms of the parameters of the prior separations and the

sort of summary statistics we would calculate anyway from the joint posterior distribution of the model actually implemented: such as posterior means and covariances. In many situations it is possible to demonstrate that these bounds between the functioning posterior and genuine posterior decrease quickly with sample size, irrespective of the likelihood - even when that likelihood is misspecified.

Under local DeRobertis (LDR) separation measures, the posterior distances between two densities is the same as the prior densities. Analogously to KL separation they also are additive under factorization so are easy to calculate or bound for most high dimensional models.

After reviewing some of the important properties of LDR in the next section we illustrate how these techniques can be used to examine analytically the robustness of inference to various forms of prior misspecification in graphical models (GMs) in Section 3.

2 Local De Robertis Separation

Let g_0 denote our *genuine prior* density and f_0 denote the *functioning prior* we actually use: usually chosen from some standard family- often products of Dirichlets - and let f_n and g_n denote their corresponding posterior densities after observing a sample $\mathbf{x}_n = (x_1, x_2, \dots x_n)$, $n \geq 1$, with *observed* sample densities $\{p_n(\mathbf{x}_n|\boldsymbol{\theta})\}_{n\geq 1}$, where $\boldsymbol{\theta} = (\theta_1, \dots, \theta_k)$. The genuine prior is unknown but we hope that it lies in some appropriate neighbourhood of f_0 so that inferences based on f_0 will be approximately right.

In many situations, because of missingness, these sample densities are typically sums of products of the conditional probabilities defining the GM so both posterior densities f_n and g_n usually have a very complicated analytic form. The functioning posterior density is therefore approximated either by drawing samples or making some algebraic computations.

Let $\Theta(n) = \{\boldsymbol{\theta} \in \Theta : p(\mathbf{x}_n|\boldsymbol{\theta}) > 0\}$, assume that $g_0(\boldsymbol{\theta})$, $f_0(\boldsymbol{\theta})$ are strictly positive and continuous on the interior of their shared support - and so uniquely defined - and assume each observed likelihood, $p_n(\mathbf{x}_n|\boldsymbol{\theta})$, $n \geq 1$ is measurable with respect to $g_0(\boldsymbol{\theta})$ and $f_0(\boldsymbol{\theta})$. From Bayes rule, for all $\boldsymbol{\theta} \in \Theta(n)$ our posterior densities $g_n(\boldsymbol{\theta}) \triangleq g(\boldsymbol{\theta}|\mathbf{x}_n)$, $f_n(\boldsymbol{\theta}) \triangleq f(\boldsymbol{\theta}|\mathbf{x}_n)$ are given by

$$\log g_n(\boldsymbol{\theta}) = \log g_0(\boldsymbol{\theta}) + \log p_n(\mathbf{x}_n|\boldsymbol{\theta}) - \log p_g(\mathbf{x}_n)$$

$$\log f_n(\boldsymbol{\theta}) = \log f_0(\boldsymbol{\theta}) + \log p_n(\mathbf{x}_n|\boldsymbol{\theta}) - \log p_f(\mathbf{x}_n)$$

where $p_g(\mathbf{x}_n) = \int_{\Theta(n)} p(\mathbf{x}_n|\boldsymbol{\theta}) g_0(\boldsymbol{\theta}) d\boldsymbol{\theta}$ and $p_f(\mathbf{x}_n) = \int_{\Theta(n)} p(\mathbf{x}_n|\boldsymbol{\theta}) f_0(\boldsymbol{\theta}) d\boldsymbol{\theta}$, whilst whenever $\boldsymbol{\theta} \in \Theta \backslash \Theta(n)$ we simply set $g_n(\boldsymbol{\theta}) = f_n(\boldsymbol{\theta}) = 0$.

For any subset $A \subseteq \Theta(n)$ let

$$d_A^L(f, g) \triangleq \sup_{\boldsymbol{\theta} \in A} \log \left\{ \frac{f(\boldsymbol{\theta})}{g(\boldsymbol{\theta})} \right\} - \inf_{\boldsymbol{\phi} \in A} \log \left\{ \frac{f(\boldsymbol{\phi})}{g(\boldsymbol{\phi})} \right\}$$

Note that this is a transparent way of measuring the discrepancy between two densities on a set A. It is non-negative, symmetric, and clearly only zero when f and g are proportional to each other - i.e. when $f(\boldsymbol{\theta}) \propto g(\boldsymbol{\theta})$, $\boldsymbol{\theta} \in A$ and $f(\boldsymbol{\phi}) \propto g(\boldsymbol{\phi})$, $\boldsymbol{\phi} \in A$. The separations have been studied when $A = \Theta(n)$ (see e.g., DeRobertis (1978); O'Hagan and Forster (2004)) but then the neighbourhoods are far too small for practical purposes. Here we focus on cases where A is chosen to be small. This allows not only the associated neighbourhoods to be realistically large but also leads to the types of strong convergence results we need.

The reason these separation measures are so important is that for *any* sequence $\{p(\mathbf{x}_n|\boldsymbol{\theta})\}_{n\geq 1}$ - however complicated -

$$d_A^L(f_n, g_n) = d_A^L(f_0, g_0) \tag{1}$$

It follows that for all sets $A \subseteq \Theta(n)$ the quality of the approximation of f_n to g_n - as measured by such a separation - is identical to the quality of the approximation of f_0 to g_0. In particular distances between two posterior densities can be calculated effortlessly from two different candidate prior densities. Unlike the functioning posterior density with missingness, the functioning prior and sometimes the genuine prior

lying in standard families and then the LDR separations can then often be expressed explicitly and always explicitly bounded. It can be shown that these separation measures are essentially the only ones with the *isoseparation property* (1) (Smith, 2007).

The fact that there are features in any prior which always endure into the posterior suggests that the priors we choose will "always" have a critical impact on inference and this will indeed be so for small sample size n. However for moderately large n the posterior f_n we calculate often places most of its mass within a set $A_n = B(\mu_n, \rho_n)$ where $B(\mu_n, \rho_n)$ denotes the open ball centred on μ_n of radius ρ_n. Write $d^L_{\Theta_0,\rho}(f,g) \triangleq \sup\{d^L_{B(\mu_n,\rho)}(f,g) : \mu_n \in \Theta_0\}$ and $d^L_{\rho}(f,g) \triangleq \sup\{d^L_{B(\mu_n,\rho)}(f,g) : \mu_n \in \Theta\}$. It has long been known that a necessary condition for robustness is that in some sense the functioning prior is "similarly smooth" to the genuine one. We therefore demand the following mild condition regulating the mutual roughness of the functioning and genuine prior. Assume that $f_0, g_0 \in \mathcal{F}(\Theta_0, M(\Theta_0), p(\Theta_0))$, where $\mathcal{F}(\Theta_0, M(\Theta_0), p(\Theta_0))$, $M(\Theta_0) < \infty, 0 < p(\Theta_0) \leq 2$ denotes the set of densities f such that for all $\boldsymbol{\theta}_0 \in \Theta_0 \subseteq \Theta$

$$\sup_{\boldsymbol{\theta},\boldsymbol{\phi} \in B(\boldsymbol{\theta}_0;\rho))} |\log f(\boldsymbol{\theta}) - \log f(\boldsymbol{\phi})| \leq$$

$$M(\Theta_0)\rho^{0.5p(\Theta_0)} \qquad (2)$$

Thus for example when $p(\Theta_0) = 2$ we demand that $\log f_0$ and $\log g_0$ both have bounded derivatives within the set Θ_0 of interest. Under these conditions Smith and Rigat (2008) show that

$$d^L_{\Theta_0,\rho}(f,g) \leq 2M(\Theta_0)\rho^{1/2p(\Theta_0)}. \qquad (3)$$

It follows that as the mass of the functioning prior converges on a ball of decreasing radius within Θ_0, $d^L_{\Theta_0,\rho}(f,g)$ converges to zero at a rate governed by the roughness parameter $p(\Theta_0)$. In particular if f, g are one dimensional densities such that $\log f$ and $\log g$ are both continuously differentiable and have derivatives bounded by M for all $\theta_0 \in \Theta_0$, then $d^L_{\rho}(f,g) \leq 2M\rho$.

Suppose the analysis of a Bayesian network is used to support decisions but the user's utility function is unknown to the modeler. If we can ensure that the *variation distance* $d_V(f_n, g_n) = \int_\Theta \mid f_n(\boldsymbol{\theta}) - g_n(\boldsymbol{\theta}) \mid d\boldsymbol{\theta}$, between f_n and g_n is small then this is sufficient to deduce that the impact of using f_n instead of g_n will not be large. For example if $d_V(f_n, g_n) < \epsilon$ then it is trivial to check that for any utility U in the class $\mathcal{U}$ of all measurable utility functions bounded below by 0 and above by 1, on a decision space $\mathcal{D}$ (Kadane and Chuang, 1978)

$$\left|\overline{U}(d^*(f_n), f_n) - \overline{U}(d^*(f_n), g_n)\right| < \varepsilon$$

for $d^*(h) = \arg\max_{d\in\mathcal{D}} \overline{U}(d,h)$ and $d \in \mathcal{D}$ where $\overline{U}(d^*(h), h) = \int_\Theta U(d, \boldsymbol{\theta}) h(\boldsymbol{\theta}) d\boldsymbol{\theta}$.

So provided that $d_V(f_n, g_n) < \varepsilon$ where $\varepsilon > 0$ is small, the consequence - measured by utility - of erroneously using f_n instead of g_n is similarly small. Conversely - unlike for the KL separation - if $d_V(f_n, g_n)$ does not tend to zero as $n \to \infty$, there is at least some utility function for which the decisions based on f_n will remain much worse than those of g_n. This has made posterior discrepancy measured through variation distance a popular choice and so is the one we focus on. In this paper we therefore investigate the conditions under which BN models are robust in this sense.

In fact the condition that the distance between the functioning and genuine prior $d^L_{B(\boldsymbol{\theta}_0;\rho)}(f_0, g_0)$ being small for small ρ is almost a sufficient condition for posterior variation distance between these densities being close for sufficiently large sample size n regardless of the value of the observed likelihood, provided that the functioning posterior concentrates its mass on a small set for large n. Below is one useful result of this type. A useful result of this type is given below.

Definition 1. Call a genuine prior g *c-rejectable* with respect to a functioning f if the ratio of marginal likelihood $\frac{p_f(\mathbf{x})}{p_g(\mathbf{x})} \geq c$.

We should believe the genuine prior will explain the data better than the functioning prior.

This in turn means that we should expect this ratio to be small and certainly not c- rejectable for a moderately large values of $c \geq 1$. Note that if the genuine prior were c - rejectable for a large c we would probably want to abandon it. For example using standard Bayesian selection techniques it would be rejected in favour of f. We need to preclude such densities from our neighbourhood.

Say density f $\Lambda-$ *tail dominates* a density g if

$$\sup_{\theta \in \Theta} \frac{g(\theta)}{f(\theta)} = \Lambda < \infty.$$

When $g(\theta)$ is bounded then this condition requires that the tail convergence of g is no faster than f. Here the prior tail dominance condition simply encourages us not to use a prior density with an overly sharp tail: a recommendation made on other grounds by for example O'Hagan and Forster (2004). The following result now holds.

Theorem 1. *If the genuine prior g_0 is not c rejectable with respect to f_0, f_0 $\Lambda-$tail dominates g_0 and $f_0, g_0 \in \mathcal{F}(\Theta_0, M(\Theta_0), p(\Theta_0))$, then for $0 < p \leq 2$*

$$d_V(f_n, g_n) \leq T_n(1, \rho_n) + 2T_n(2, \rho_n) \qquad (4)$$

where

$$T_n(1, \rho_n) = \exp d^L_{\mu, \rho_n}(f, g) - 1 \leq \exp\left\{2M\rho_n^{p/2}\right\} - 1$$

and $T_n(2, \rho_n) = (1 + c\Lambda)\alpha_n(\rho_n)$, where $\alpha_n(\rho_n) = F_n(\theta \notin B(\theta_0, \rho_n))$ and $F_n(.)$ stands for the cumulative distribution function of θ .

Proof. See Appendix in Smith (2007). □

It is usually easy to bound $T_n(2, \rho_n)$ explicitly using Chebychev type inequalities (see Smith, 2007 for more details). One useful bound, sufficient for our present context, is given below. It assumes that we can calculate or approximate well the posterior means and variances of the vector of parameters under the functioning prior. These posterior summaries are routinely calculated in most Bayesian analyses.

Example 1. Let $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k)$ and $\mu_{j,n}, \sigma^2_{jj,n}$ denote, respectively, the mean and variance of θ_j, $1 \leq j \leq k$ under the functioning posterior density f_n. Then Tong (1980, p153) proves that, writing $\boldsymbol{\mu}_n = (\mu_{1,n}, \mu_{2,n}, \ldots \mu_{k,n})$

$$\begin{aligned} & F_n\left(\boldsymbol{\theta} \in B(\boldsymbol{\mu}_n; \rho_n)\right) \\ \geq \; & F_n\left[\bigcap_{j=1}^{k} \left\{|\theta_j - \mu_{j,n}| \leq \sqrt{k}\rho_n\right\}\right] \\ \geq \; & 1 - k\rho_n^{-2} \sum_{j=1}^{k} \sigma^2_{jj,n} \end{aligned}$$

so that $F_n\left(\boldsymbol{\theta} \notin B(\boldsymbol{\mu}_n; \rho_n)\right) \leq k\rho_n^{-2} \sum_{j=1}^{k} \sigma^2_{jj,n}$ implying

$$T_n(2, \rho_n) \leq c\Lambda\sigma_n^2\rho_n^{-2},$$

where $\sigma_n^2 = k \max_{1 \leq j \leq k} \sigma^2_{j,n}$. In many cases we can show that $\sigma_n^2 \leq n^{-1}\sigma^2$ for some value σ^2. Note that this gives an explicit upper bound on $T_n(2, \rho_n)$ which tends to zero provided ρ_n is chosen so that $\rho_n^2 \leq n^r\rho$ where $0 < r < 1$.

For a fixed (small) ρ, provided σ_n^2 is sufficiently small $d_V(f_n, g_n)$ will also be small. Indeed when $p = 2$ it will tend to zero at any rate slower than the rate σ_n^2 converges to zero. The other component of our bound $T_n(1, \rho_n)$ can also be calculated or bounded for most standard multivariate distributions. A simple illustration of this bound, where both the functioning prior and genuine prior are drawn from the same family, is given below.

Example 2. Let $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k)$ $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$, $\theta_i, \alpha_i > 0, \sum_{i=1}^{k} \theta_i = 1$ - so that Θ is the k simplex. Let the two prior densities $f_0(\boldsymbol{\theta} \mid \boldsymbol{\alpha}_f)$ and $g_0(\boldsymbol{\theta} \mid \boldsymbol{\alpha}_g)$ be *Dirichlet* so that

$$f_0(\boldsymbol{\theta} \mid \boldsymbol{\alpha}_f) \propto \prod_{i=1}^{k} \theta_i^{\alpha_{i,f}-1}, \;\; g_0(\boldsymbol{\theta} \mid \boldsymbol{\alpha}_g) \propto \prod_{i=1}^{k} \theta_i^{\alpha_{i,g}-1}$$

Let $\boldsymbol{\mu}_n = (\mu_{1,n}, \mu_{2,n}, \ldots, \mu_{k,n})$ denote the mean of the functioning posterior density f_n. Then it can be easily checked that if $\rho_n < \mu_n^0 = \min\{\mu_{i,n} : 1 \leq i \leq k\}$, then $d^L_{\boldsymbol{\mu}_n; \rho_n}(f_0, g_0)$ is bounded above by

$$\sum_{i=1}^{k} |\alpha_{i,f} - \alpha_{i,g}| \left\{\log(\mu_{i,n} + \rho_n) - \log(\mu_{i,n} - \rho_n)\right\}$$

$$\leq 2k\rho_n \left(\mu_n^0 - \rho_n\right)^{-1} \overline{\alpha}(f_0, g_0)$$

where $\overline{\alpha}(f_0, g_0) = k^{-1} \sum_{i=1}^{k} |\alpha_{i,f} - \alpha_{i,g}|$ is the average distance between the hyperparameters of the functioning and genuine priors. So $T_n(1, \rho_n)$ is uniformly bounded whenever $\boldsymbol{\mu}_n$ remains in a given fixed closed interval Θ_0 for all n and converges approximately linearly in n. Note that in the cases above, provided we ensure $\rho_n^2 \leq n^r \rho$, $0 < r < 1$ then both $T_n(1, \rho_n)$ and $T_n(2, \rho_n)$ - and hence $d_V(f_n, g_n)$ - tends to zero. However if f_n tends to concentrate its mass on the boundary of Θ near one of the cell probabilities being zero, then even when the average distance $\overline{\alpha}(f, g)$ between the hyperparameters of the priors are small, it can be shown that at least some likelihoods will force the variation distance between the posterior densities to stay large for increasing ρ_n. See Smith (2007) for a proof and an explicit example of this phenomenon. Typically the smaller the probability the slower any convergence in variation distance will be.

Example 3. Sometimes it is convenient, particularly with covariate information, to smoothly transform a vector of probabilities. One commonly used transformation in BNs is the logistic transformation (Spiegelhalter and Laritzen, 1990). Like the variation distance the LDR is invariant to diffeomorphic transformations like this one. When the learning has proceeded on this transformed scale it is often expedient to use this scale directly in the use of Theorem 1. Note that under the logistic transformation we can identify the problem area of inference in the example above - i.e. where the posterior concentrates near a zero in one of the component probabilities, corresponds exactly to the well known sensitivity to tail behaviour when outliers are observed (O'Hagan (1979); Andrade and O'Hagan (2006)). Any family of distributions on the transformed scale having sub-exponential tails - for example multivariate t-distribution has better robustness properties both in term of the LDR and the tail domination condition above than super-exponential tails families - like the Gaussian, and should be preferred in this context (O'Hagan and Forster, 2004).

Of course the usual priors in discrete GMs are typically *products* of many such Dirichlet densities. However our local separation for these products is similarly easily explicitly bounded: see below.

It is interesting to note that lower bounds on variation distances can calculated given that $d^L_{\boldsymbol{\mu}_n;\rho_n}(f_0, g_0)$ stay unbounded above as $n \to \infty$. Thus Smith (2007) show that whenever $d^L_{\boldsymbol{\mu}_n;\rho_n}(f_0, g_0)$ does not converge to zero as $\rho_n \to 0$, in general. Of course our genuine prior g_0 need not be Dirichlet even if the functioning prior is. However, the general conditions above ensure that except when posterior distribution of a single vector of probabilities under the functioning prior tend to zero in some component or unless the prior we should use is much rougher (or smoother) than f_0 with large n we will obtain approximately the right answer in the sense described above.

Note that if two priors are close with respect to LDRs, even when the likelihood is inconsistent with the data, the functioning posterior distribution nevertheless will tend to provide a good approximation of the genuine posterior as the functioning posterior concentrates. All similar priors will give similar (if possibly erroneous) posterior densities.

We now proceed to investigate the properties of $d^L_{\boldsymbol{\mu}_n;\rho_n}(f_0, g_0)$ for graphical models.

3 Isoseparation and BN's

3.1 Some General Results for Multivariate BN's

We begin with some general comments about multivariate robustness.

In Smith and Rigat (2008) it is proved that if $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ and $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$ are two candidate parameter values in $\Theta = \Theta_1 \times \Theta_2$ where $\boldsymbol{\theta}_1, \boldsymbol{\phi}_1 \in \Theta_1$ and $\boldsymbol{\theta}_2, \boldsymbol{\phi}_2 \in \Theta_2$, where the joint densities $f(\boldsymbol{\theta})$, $g(\boldsymbol{\theta})$ are continuous in Θ and $f_1(\boldsymbol{\theta}_1), g_1(\boldsymbol{\theta}_1)$ represent the marginal densities on Θ_1 of the two joint densities $f(\boldsymbol{\theta})$ and $g(\boldsymbol{\theta})$ respectively, then

$$d^L_{A_1}(f_1, g_1) \leq d^L_A(f, g) \tag{5}$$

where $A_1 = \{\boldsymbol{\theta}_1 : \boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \in A$ for all $\boldsymbol{\theta}_2 \in B \subset \Theta_2$ for some open set B in $\Theta_2\}$. So in particular marginal densities are never more separated than their joint densities. Thus if we are interested only in particular margins of the probabilities in a BN and we can show that the functioning prior converges on that margin, then even if the model is unidentified provided $f_0, g_0 \in \mathcal{F}(\Theta_0, M(\Theta_0), p(\Theta_0))$, we will still be able to assert - using an argument exactly analogous to that in the proof of Theorem 1 that with large n the functioning prior will be a good surrogate for the genuine one. This is important since we know that BNs with interior systematically hidden variables are unidentified. However if our utility function is a function only of the manifest variables we can ensure that the variation distance between two posterior marginal densities $f_{1,n}$ $g_{1,n}$ become increasing close - usually at a rate of at least $\sqrt[3]{n}$ - in variation. So in such a case lack of robustness only exists on prior specifications of functions of probabilities of the conditional distributions of the hidden variables conditional on the manifest variables.

Next we note that the usual convention is to use BNs whose probabilities all exhibit prior local and global independence (LGI). Immediately from the definition of $d_A^L(f, g)$ if $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots \boldsymbol{\theta}_k)$ with functioning prior $f(\boldsymbol{\theta})$ and genuine prior $g(\boldsymbol{\theta})$ both with the property that subvectors $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots \boldsymbol{\theta}_k\}$ of parameters are mutually independent so that

$$f(\boldsymbol{\theta}) = \prod_{i=1}^{k} f_i(\boldsymbol{\theta}_i), \quad g(\boldsymbol{\theta}) = \prod_{i=1}^{k} g_i(\boldsymbol{\theta}_i)$$

where $f_i(\boldsymbol{\theta}_i)$ $(g_i(\boldsymbol{\theta}_i))$ are the functioning (genuine) margin on $\boldsymbol{\theta}_i$, $1 \leq i \leq k$, then

$$d_A^L(f, g) = \sum_{i=1}^{k} d_{A_i}^L(f_i, g_i) \tag{6}$$

It follows that - all other things being equal - our local prior distances grow linearly with the number of parameters needed to specify a BN. In particular models encoding more conditional independences are intrinsically more stable and the effects of possibly erroneous prior information will endure longer than more complex models encoding less conditional independences. It has long been known that Bayesian selection methods, for example based on Bayes Factors automatically select simpler models when they provide similar explanation of the observed data than more complex models. But here we have a complementary point. The choice of the complex model will tend to give less reliable posteriors if we are not absolutely sure of our priors.

Example 4. Suppose a discrete BN G on $\{X_1, X_2, \ldots, X_m\}$ where X_i has t levels and parents Pa_i, taking on s_i different parent configurations, $1 \leq i \leq m$. Make the common assumption that our genuine and functioning prior both exhibits LGI: i.e. all $s = \prod_{i=1}^{m} s_i$ parameter vectors $\boldsymbol{\theta}_i | pa_i$ are mutually independent under both f and g. If we believe the LDR separation between the s component densities of the functioning and genuine prior is δ_A then $d_A^L(f, g) = s\delta_A$. Note that the quality of the approximation will depend on the number of parent configurations in the model. Thus if G^1 has all components independent, G^2 is a tree, G^3 is complete and f^j, g^j are the prior densities under G^j, $j = 1, 2, 3$ then

$$d_A^L(f^1, g^1) = m\delta_A, \;\; d_A^L(f^2, g^2) = \{mt - t + 1\}\,\delta_A$$

$$d_A^L(f^3, g^3) = \{t^m - 1\}\,\{t - 1\}^{-1}\,\delta_A$$

The last most general separation bound increases exponentially with m. By (5) this in turn implies that BN's containing a large clique are most unreliable in the sense that data size has to be enormous before we can be confident our inferences are approximately reliable in the sense measured by LDR. Note that in this setting the bound given by our first example on the second component $T_n(2, \rho_n)$ in our theorem is a function of the mean and variances of the component vectors of probabilities (or in some analyses their logistic transform). These are routinely sampled anyway so good estimates can just be plugged in our formula and together with the bounds above this provides explicit operational uncertainty bounds on our variation distances.

Example 5. If the BN is decomposable with cliques $C[j]$, $j = 1, 2, \ldots, m$ then if we require LGI to hold in all Markov equivalent graphs then it is proved that the joint distribution of the clique probabilities on the vector of probability tables over each clique must have a Dirichlet distribution (with consistent distributions over separators). This in turn implies all conditional probabilities used in a BN will also be Dirichlet for both the genuine and functioning prior allowing us to calculate explicit expressions for distances between components. Here we note again that prior distances are expressed through a Euclidean distance on the hyperparameters of the genuine and functioning prior then posterior variation instabilities can occur in the limit only if our posterior density concentrates near zero on some component. Although this phenomenon is unusual for many likelihoods where components are missing at random this is not the case when some components are systematically missing (Smith and Croft, 2003). Indeed when estimating probabilities on phylogenetic trees where only the root and leaf nodes are observed and all probabilities are free it is the norm in practice to find the distribution of at least some of the internal hidden nodes concentrating near zero on some of the probabilities. In these cases, whilst it can be shown that the estimates of the marginal manifest probabilities are usually stable under large samples and the prior may well have a large effect on the inferences about the internal explanatory probabilities, even when the probabilities are identifiable and samples are very large. Unfortunately these probabilities are often the ones of scientific interest!

3.2 Sensitivity to Departures in Parameter Independence

Although LGI is a useful expedient, if a prior is elicited using contextual information - as it should be- systematic biases in the elicitation processes due to poor calibration or selection bias will break these assumptions dramatically. The issue then is to what extent using the assumption of LGI matters. One possible extension away from LGI that naturally occurs under selection biases is for the vector of probabilities in the problem to mirror the dependence structure of the BN G. A special case of this is when we drop the local independence assumption. So suppose a functioning prior $f(\boldsymbol{\theta})$ and a genuine prior $g(\boldsymbol{\theta})$ where $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots \theta_k) \in \Theta = \Theta_1 \times \Theta_2 \times \ldots \times \Theta_k$ are both constrained to respect the same factorisation

$$f(\boldsymbol{\theta}) = f(\theta_1) \prod_{i=2}^{k} f_{i|.}(\theta_i \mid \boldsymbol{\theta}_{pa_i})$$

$$g(\boldsymbol{\theta}) = g(\theta_1) \prod_{i=2}^{k} g_{i|.}(\theta_i \mid \boldsymbol{\theta}_{pa_i}),$$

where for $2 \leq i \leq k$, the parents $\boldsymbol{\theta}_{pa_i}$ of θ_i is a subvector of $(\theta_1, \theta_2, \ldots \theta_{i-1})$. Write $\boldsymbol{\theta}[1] = \theta_1 \in \Theta[1] = \Theta_1$ and $\boldsymbol{\theta}[i] = (\theta_i, \boldsymbol{\theta}_{pa_i}) \in \Theta[i]$, $2 \leq i \leq k$. Let $A = A[1] \times A[2] \times \ldots \times A[k] \subseteq \Theta$ where $A[i] \subseteq \Theta[i]$, $1 \leq i \leq k$. Then it is straightforward to show that $d_A^L(f, g) \leq \sum_{i=2}^{k} d_{A[i]}^L(f_{[i]}, g_{[i]})$ where $f_{A[i]}, g_{A[i]}$ are respectively the margin of f and g on the space $\Theta[i]$ of the i^{th} variable and its parents (Smith, 2007). Note therefore that our local separations increase no faster than linearly in the number of probabilities. It is natural to set these bounds so that they are functionally independently of the particular parent configuration $\boldsymbol{\theta}_{pa_i}$.

Definition 2. Say the neighbourhood $\mathcal{N}(f)$ of $f(\boldsymbol{\theta}) = f(\theta_1) \prod_{i=2}^{k} f_{i|.}(\theta_i \mid \boldsymbol{\theta}_{pa_i})$ is *uniformly A uncertain* if $g \in \mathcal{N}(f)$ respect the same factorisation as f and

$$\sup_{g \in \mathcal{G}(f)} \sup_{\theta_i, \phi_i \in A[i]} \log \left\{ \frac{f_{i|.}(\theta_i, \boldsymbol{\theta}_{pa_i})\, g_{i|.}((\phi_i, \boldsymbol{\theta}_{pa_i})}{g_{i|.}(\theta_i, \boldsymbol{\theta}_{pa_i})\, f_{i|.}(\phi_i, \boldsymbol{\theta}_{pa_i})} \right\}$$

is not a function of $\boldsymbol{\theta}_{pa_i}$ $2 \leq i \leq n$.

If we believe the genuine prior $g \in \mathcal{G}(f)$ is uniformly A uncertain then we can write $d_A^L(f, g) = \sum_{i=1}^{k} d_{A[i]}^{L*}(f_{i|.}, g_{i|.})$ (see Smith, 2007).

The separation between the joint densities f and g is then simply the sum of the separation between its component conditionals $f_{i|.}$ and $g_{i|.}$, $1 \leq i \leq k$. So in particular we can calculate bounds for the joint density of the genuine posterior from prior smoothness conditions on

each of the genuine and functioning conditionals and parameters of the posterior. Notice that these bounds will apply *even* when the likelihood destroys the factorisation of the prior. So the critical property we assume here is the fact that we believe a priori that f respects the same factorisation as g. If we learn the value of $\boldsymbol{\theta}(I) = \{\theta_i : i \in I\}$ where I is some index set then the separation between the densities reduces to

$$d_A^L(f(.|\boldsymbol{\theta}(I)), g(.|\boldsymbol{\theta}(I))) = \sum_{i \notin I} d_{A[i]}^{L*}(f_{i|.}, g_{i|.})$$

There is therefore a degree of stability to deviations in parameter independence assumptions.

Finally consider the general case where the hyperprior is totally general but the modeler believes that the dependence between parameters has been caused by the expert first assuming all component probabilities as mutually independent and then observing a particular data set $\mathbf{y}$ with sample mass function $q(\mathbf{y}|\boldsymbol{\theta}) > 0$ and forming her new dependent posterior. If we assume that deviation in this process is only caused by the misspecification of the initial independence prior then by the isoseparation property, the LDR discrepancy between genuine and functioning prior should be set at the same deviation parameters as the independence priors. So on this strong assumption we regain the stability existing under LGI.

4 Discussion

For any BNs whose densities factorise, the LDR separations are a valuable way of understanding exactly what forces the final posterior inferences. Robustness under large n will typically exist for sparse graphs with no component probabilities close to zero. On the other hand graphical models with many boundary probabilities and/or a large number of edges will exhibit enduring large approximation errors measured in total variation distance. This gives yet another reason why restricting inference with BN's to graphs with only a small number of edges is a good idea.

We note that the same techniques can be used to study inference in continuous and mixed BN's and also for all other GMs encoding a single factorization. We are currently implementing these techniques and the bounds appear to provide genuinely helpful supplementary diagnostic information to what is often a complex estimation exercise.

References

J. A. A. Andrade and Anthony O'Hagan. 2006. Bayesian robustness modelling using regularly varying distributions. *Bayesian Analysis*, 1:169–188.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen and D. J. Spiegelhalter. 2000. *Probabilistic Networks and Expert Systems*. Springer Verlag.

L. DeRobertis. 1978. *The use of partial prior knowledge in Bayesian inference*. Ph.D. dissertation, Yale Univ.

P. Gustafson and L. Wasserman. 1995. Local sensitivity diagnostics for Bayesiain inference. *Annals Statist*, 23:2153–2167.

J. B. Kadane and D. T. Chuang. 1978. Stable decision problems. *Ann. Statist*, 6:1095–1111.

A. O'Hagan. 1979. On outlier rejection phenomena in Bayesian inference. *J. R. Statist. Soc. B*, 41:358–367.

A. O'Hagan and J. Forster. 2004. *Bayesian Inference*. Kendall's Advanced Theory of Statistics, Arnold.

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

J. Q. Smith. 2007. Local Robustness of Bayesian Parametric Inference and Observed Likelihoods. *CRiSM Res Rep*, 07-08.

J. Q. Smith and J. Croft. 2003. Bayesian networks for discrete multivariae data: an algebraic approach to inference. *J of Multivariate Analysis*, 84(2):387–402.

J. Q. Smith and F. Rigat. 2008. Isoseparation and Robustness in Finitre Parameter Bayesian Inference. *CRiSM Res Rep*, 07-22.

Y. L. Tong. 1980. *Probability Inequalities in Multivariate Distributions*. Academic Press New York.

Eliciting expert beliefs on the structure of a Bayesian Network

Federico M. Stefanini
Department of Statistics 'G.Parenti'
University of Florence, viale Morgagni 59, 50134 Firenze, Italy

Abstract

The elicitation of prior beliefs about the structure of a Bayesian Network is a formal step of full-Bayesian structural learning which offers the opportunity of exploiting the knowledge accumulated by an expert of the problem domain over years of research in a quantitative way. Motivating applications include molecular biomarkers in gene expression or protein assays, where the use of prior information is often suggested as a promising approach to face the curse of dimensionality. In this paper a general formalization based on propositions describing network features is developed which comprises issues like anchoring and revision. An algorithm is described to estimate the number of structures bearing a-priori relevant features in problem domains characterized by a large number of nodes.

1 Introduction

The structure of a Bayesian Network (BN) and its parameters are in many cases unknown or affected by substantial uncertainty, therefore network learning is performed on the basis of collected data. A prior distribution over the space of structures is a formal ingredient of the Bayesian paradigm. Nevertheless, the elicitation of expert's prior information on network's structure suffers a major limitation due to the super-exponential increase of structures to be considered which becomes critical for five or more random variables. Despite the above mentioned difficulties, there is a wide agreement on the possibility of mitigating the 'curse of dimensionality' occurring in many applied fields by using prior information elicited from experts of the problem domain.

Several approaches have been proposed to define a prior distribution on the set of DAGs for a fixed set of variables. The early work of Buntine (Buntine, 1991) is based on a total ordering of nodes and a full specification of beliefs for each edge which could join pairs of nodes in a DAG. The collection of nodes which precedes a given node v is known given the order relation, therefore the probability of a given parent set Π_v of node v may be calculated as the product of probability for events of type 'there is edge $y \to v$' or 'there is not an edge $y \to v$', for each y preceding v. The subjective probability elicited from an expert about structure B_s is defined as the product of probability values for each parent set marginally considered. In the seminal paper of Heckerman (Heckerman et al., 1995) a prior network, $\mathcal{B}_{sc}$, is elicited and compared with candidate networks so that a high degree of belief is assigned to structures closely resembling to the prior network. The number δ_i of different nodes in the parent set of node v_i is calculated for each node to quantify the overall degree of dissimilarity $\delta = \sum_i \delta_i$. Given an elicited hyperparameter $0 < k < 1$, the prior distribution is proportional to k^{δ}. Castelo and Siebes first addressed the issue of partial prior knowledge and they also provided automatic rules to obtain a full prior for a Bayesian network (Castelo and Siebes, 2000). Recent contributions include the development of an informed score function based on the BDe metric (Mascherini and Stefanini, 2007). The use of several types of restrictions to code expert knowledge in structural learning of BNs has been investigated by (de Campos and Castellano, 2007), who also particularized the approach to the local search and to the PC learning algorithms.

This paper is motivated by the need of elicit-

ing beliefs in a more general setup, e.g. avoiding both the a-priori independence among parent sets and the specification of a prior network. A formal approach is developed with the aim of supporting researchers of applied fields in the elicitation and revision of causal and probabilistic beliefs. An algorithm is described which is useful in problem domains characterized by a large number of nodes. A simple case study is presented to illustrate the approach. The key idea of this paper is that in large spaces of structures, elicitation may deal with a limited number of network features.

2 Material and Methods

2.1 Bayesian networks

A graph $\mathcal{G}$ is a pair (V, E) where $V = \{v_1, v_2, \ldots, v_K\}$ is a finite set of nodes and $E \subset V \times V$ is the set of edges. The set E represents the structure of the graph because it defines which nodes are linked by an edge and if such edge is oriented (arrow) or not (undirected). If $(v_i, v_j) \in E$ but $(v_j, v_i) \notin E$ then the ordered pair corresponds to the oriented edge $v_i \rightarrow v_j$. The set of nodes originating oriented edges that enter into node v_j is called parents set, denoted as $pa(v_j)$. In a directed graph all edges are oriented. In a directed graph without cycles a tour following the direction of oriented edges never visits the same node two times. A directed graph without cycles is called Directed Acyclic Graph (DAG). An auxiliary random variable Z is introduced to map the set of DAGs for a fixed V to the set of natural numbers. It follows that a structure E is associated to an arbitrary number z in the set $\Omega_Z = \{1, 2, \ldots, n_z\}$, the sample space of Z.

The joint probability distribution of random variables indexed in V, the random vector $X_{v_1,\ldots,v_K}$, is Markov with respect to a DAG $\mathcal{G}$ if the following factorization holds:

$$p(x_{v_1}, x_{v_2}, \ldots, x_{v_K}) = \prod_{v_i \in V} p(x_{v_i} \mid x_{pa(v_i)})$$

where $x_{pa(v_i)}$ is random vector made by variables whose labels belong to the parents set of v_i. The lack of an arrow from v_i to v_j means irrelevance of X_{v_i} in predicting X_{v_j} if all random variables defined by the parent set have been observed, i.e. it is an instance of conditional independence. More general conditional independence statements may be derived by means of the D-separation theorem, or equivalently separation theorems on moralized graphs (Cowell et al., 1999). Under the stronger Markov Causal Assumption a DAG represents relations among variables which are stable under external manipulation (intervention) of a subset of them, so that causal effects may be in principle estimated.

Structural learning of a BN amounts to process a database $\mathcal{D} = \{d_1, d_2, \ldots, d_{n_d}\}$ of n_d conditionally independent realizations of the random vector $X_{v_1,v_2,\ldots,v_K}$ to infer the conditional independence relations existing in the joint distribution of the random vector. Following the Bayesian approach to inference, the joint probability distribution of $\mathcal{D}$ and network's unknowns given the context ξ is

$$p(\mathcal{D}, \theta, z \mid \xi) = p(\mathcal{D} \mid \theta, z, \xi) \cdot p(\theta \mid z, \xi) \cdot p(z \mid \xi),$$

where $\theta = (\theta_{v_1,pa(v_1)}, \ldots, \theta_{v_K,pa(v_K)})$ are vectors of parameters which appear in the conditional probability distributions of each pair X_{v_i}, $X_{pa(v_i)}$. The likelihood function $p(\mathcal{D} \mid \theta, z, \xi)$ is a product of multinomials and the degree of belief about elements of θ is often expressed as a product of Dirichlet probability density functions (Heckerman et al., 1995). The probability mass function $p(z \mid \xi)$ captures the expert's degree of belief about the unknown structure of a BN.

2.2 Expert's degree of belief about network features

In typical problem domains, we expect that an expert is willing to believe more on candidate structures showing some important features which are a-priori plausible.

Definition 1 (Features)**.** Network features $\{\mathcal{P}_1, \mathcal{P}_2, \ldots\}$ are propositions qualifying graphs defined on a fixed set of nodes V. Given a structure z, a proposition $\mathcal{P}_i(z)$ is either true or false.

Among the examples of features we have: $\mathcal{P}_1$ = 'is an ancestor of v_4', $\mathcal{P}_2$ = 'maximum

cardinality of parents $\leq 2\ \forall v \in V$', $\mathcal{P}_3$ = 'maximum cardinality of children $\leq 2\ \forall v \in V$', $\mathcal{P}_4$ = 'node v_3 is neighbor of v_7', $\mathcal{P}_5$ = 'variable X_{v_2} is an immediate cause of X_{v_7}'. Features may accommodate both probabilistic and causal beliefs according to the choice of suitable propositions and context.

Expert's belief is typically elicited through several network features which may or may not hold at once. Nevertheless the straight specification of $p(z \mid \mathcal{P}_1, \mathcal{P}_2, \ldots)$ may be difficult for a general collection of statements due to relations which might exist among propositions. A collection of features may be organized into a basis for the elicitation by defining canonical features.

Definition 2 (Canonical feature)**.** Let $\mathcal{R} = \{\mathcal{P}_i : i = 1, \ldots, n_f\}$ be the set of reference features selected by an expert for the elicitation. A canonical feature $\mathcal{F}_j, j \in \mathcal{J}$, is a conjunction $\bigwedge_{i=1}^{n_f} \widetilde{\mathcal{P}}_i$ where $\widetilde{\mathcal{P}}_i$ is a proposition chosen between $\mathcal{P}_i$ and its negation $\neg\mathcal{P}_i$.

A convenient index set is $\mathcal{J} = \{(\overline{1}, \ldots, \overline{n_f}), \ldots, (1, \ldots, n_f)\}$ so that the configuration of features in $\mathcal{R}$ which generate a canonical feature $\mathcal{F}_j$, $j \in \mathcal{J}$, is self-evident. Note that a canonical feature is defined in a context ξ which includes a fixed collection of random variables.

Definition 3 (Elicitation basis)**.** A canonical reference set $\mathcal{F} = \{\mathcal{F}_j : j \in J\}$ built on reference set $\mathcal{R}$ is the collection of canonical features defined by all the possible conjunctions in Definition 2. It is a basis for the elicitation.

A canonical reference set induces a partition on the set of structures, as stated in the proposition below:

Proposition 1 (Canonical partition)**.** *Let $\mathcal{F}$ be an elicitation basis on a fixed $\mathcal{R}$. Let $\mathcal{C}_j = \{z : \mathcal{F}_j(z)\}$ be the set of structures satisfying the assertion $\mathcal{F}_j \in \mathcal{F}$. The set $\mathcal{C} = \{\mathcal{C}_j : j \in \mathcal{J}\}$ is a partition of the set of graphs built on a fixed collection of nodes V.*

Proof. By definition, canonical features are incompatible propositions. □

We are now in the position of eliciting a quantitative preference relation on such a partition.

Definition 4 (Preference on $\mathcal{F}$)**.** Let $\mathcal{F}$ be a canonical reference set. A preference relation $\mathcal{U}$ induces an order on $\mathcal{F}$ and the resulting 'precede' and 'succeed' relations are respectively indicated as $\prec$ and $\succ$. If $\mathcal{U}$ determines a partial ordering of features then $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_e, \ldots, \mathcal{E}_{n_e}\}$ is the induced partition of $\mathcal{F}$ into equivalence classes, with $\mathcal{E}_e$ a generic member of partition $\mathcal{E}$, n_e the total number of equivalence classes and $\mathcal{F}_{[e]}$ a generic member of $\mathcal{E}_e$.

The preference relation $\mathcal{U}$ is not necessarily a strict ordering because different canonical features may be equally plausible for the expert. A non-trivial elicitation basis $\mathcal{F}$, contains at least two distinct elements $\mathcal{F}_L$ and $\mathcal{F}_U$, with $\mathcal{F}_L \prec \mathcal{F}_U$, that respectively precedes and succeeds other canonical features. Therefore degrees of belief satisfy the inequality: $P[\mathcal{F}_L \mid \mathcal{U}, \xi] < P[\mathcal{F}_U \mid \mathcal{U}, \xi]$. A generic canonical feature $\mathcal{F}_j, j \in \mathcal{J}$, does not succeed to $\mathcal{F}_U$ and it does not precedes $\mathcal{F}_L$, that is $\mathcal{F}_L \preceq \mathcal{F}_j \preceq \mathcal{F}_U$, therefore the degree of belief satisfies:

$$P[\mathcal{F}_L \mid \mathcal{U}, \xi] \leq P[\mathcal{F}_j \mid \mathcal{U}, \xi] \leq P[\mathcal{F}_U \mid \mathcal{U}, \xi].$$

Note that if $\mathcal{F}_{j'} \preceq \mathcal{F}_{j''}$ and $\mathcal{F}_{j'} \succeq \mathcal{F}_{j''}$ both holds for two canonical features $\mathcal{F}_{j'}$ and $\mathcal{F}_{j''}$, then they belong to the same equivalence class, namely $\mathcal{F}_{j'} \sim \mathcal{F}_{j''}$ induced by $\mathcal{U}$.

The numerical assignment of degrees of belief is here performed using conditional odds.

Definition 5 (Conditional odds)**.** Let $\mathcal{F}_a, \mathcal{F}_b$ two canonical features and $\mathcal{U}$ a preference relation on $\mathcal{F}$. Conditional odds of $\mathcal{F}_a$ against $\mathcal{F}_b$ given $\mathcal{U}, \xi$ are:

$$\omega_{a,b} = \frac{P[\mathcal{F}_a \mid \mathcal{U}, \xi]}{P[\mathcal{F}_b \mid \mathcal{U}, \xi]} \tag{1}$$

with $\omega_{a,b} \geq 0$.

It follows from (1) that the numerical assignment for two features $\mathcal{F}_{j'} \sim \mathcal{F}_{j''}$ belonging to the same equivalence class is $\omega_{j',j''} = 1.0$. The direct numerical assignment of the degree of belief for pairs of features belonging to distinct equivalence classes $\mathcal{E}_a$ and $\mathcal{E}_b$ exploits an auxiliary experiment, here a hypothetical random

draw of one ball from an urn which contains α_r red balls and α_w white balls, with $\alpha_r + \alpha_w$ conveniently set to 100 or more. Given two features $\mathcal{F}_{[a]}$ and $\mathcal{F}_{[b]}$, the number of white and of red balls in the urn has to be changed by the expert up to the point in which the odds associated to the proposition 'the ball drawn from the urn is white' are equal to conditional odds of $\mathcal{F}_{[a]}$ against $\mathcal{F}_{[b]}$: $\omega_{a,b} = \frac{\alpha_w}{\alpha_r}$.

Proposition 2 (Complete minimal ensemble)**.** *Let $\mathcal{U}$ be an order relation on $\mathcal{F}$ and $\mathcal{E}$ the induced partition into equivalence classes. An ensemble is a collection of conditional odds $\{\omega_{a,b}\}$ elicited from the expert. The ensemble is complete and minimal if it contains $n_e - 1$ odds values between pairs of features belonging to distinct equivalence classes, so that at least one feature is taken from each equivalence class in $\mathcal{E}$.*

Proof. The ensemble is complete because a probability distribution on $\mathcal{F}$ is obtained by transformation of elicited odds, that is $\sum_{j\in\mathcal{J}} P[\mathcal{F}_j \mid \mathcal{U}, \xi] = 1$ and $P[\mathcal{F}_j \mid \mathcal{U}, \xi] \geq 0$. The ensemble is minimal because its size can not be further reduced without compromising the full specification of a probability distribution on $\mathcal{F}$. □

The assignment of conditional odds has to be performed according to the order induced by $\mathcal{U}$ in units of subjective probability.

Two structures z_1 and z_2 may belong to the same equivalence class $\mathcal{C}_j$ and in this case they are on equal footing for what concerns expert's prior information. The probability $P[Z = z \mid \mathcal{F}_j, \xi]$ represents the expert degree of belief about the proposition: 'the unknown structure z is one of those structures characterized by $\mathcal{F}_j$'.

Proposition 3 (Beliefs on Z)**.** *Given the canonical partition $\mathcal{C}$ induced by an elicitation basis $\mathcal{F}$, the probability mass function $p(z \mid \xi)$ is given by:*

$$p(z \mid \mathcal{U}, \xi) = \frac{1}{n_{j(z)}} \cdot P[\mathcal{F}_{j(z)} \mid \mathcal{U}, \xi] \qquad (2)$$

where $j(z)$ is the element of the canonical partition in which z is located, and with $n_{j(z)}$ the cardinality of such subset.

Proof. The starting factorization is:

$$p(z \mid \mathcal{U}, \xi) = \sum_{j\in\mathcal{J}} P[Z = z \mid \mathcal{F}_j, \xi] \cdot P[\mathcal{F}_j \mid \mathcal{U}, \xi]$$

but $P[Z = z \mid \mathcal{F}_j, \xi]$ is null for all but one conditioning feature, say $\mathcal{F}_{j(z)}$. Moreover, under indifference among members within class $\mathcal{C}_{j(z)}$ the probability $P[Z = z \mid \mathcal{F}_{j(z)}, \xi]$ is one over $n_{j(z)}$, the cardinality of such equivalence class. □

2.3 The elicitation of $p(z \mid \mathcal{U}, \xi)$

An elicitation basis is a general object, nevertheless it is convenient to describe some practical details both to support algorithms formulation and to prepare the expert to variations which also depend on the amount of information being elicited. The conjunction of two or more incompatible features, like $\mathcal{P}_{i'}$ = 'has $v_i \to v_j$' and $\mathcal{P}_{i''}$ = 'has $v_j \to v_i$', determines a canonical feature which is indeed false for DAGs. Therefore the probability of $\mathcal{P}_{i'} \wedge \mathcal{P}_{i''}$ is null. Similar remarks hold if a feature implies another feature, say $\mathcal{P}_2 \Rightarrow \mathcal{P}_1$. In such case the degree of belief in the conjunction $\neg\mathcal{P}_1 \wedge \mathcal{P}_2$ should be zero.

Substantial prior information in the problem domain may result in a narrow partition, $n_{j(z)} = 1, \forall j \in \mathcal{J}$, and the burden of assessment is equivalent to the one-by-one elicitation of beliefs on structures. Nevertheless, in large spaces of structures it is likely that the elicitation brings to coarse partitions in which $n_j \gg 1$, and the number of DAGs belonging to an equivalence class may be hard to assess.

An approximated solution to the counting problem may be obtained by simulation. The core of our algorithm is defined in (Ide et al., 2002) who build a Markov Chain (MC) that at convergence provides a DAG uniformly sampled from the space of all DAGs on a fixed set of nodes V. We extended the algorithm described in (Ide et al., 2002, Algorithm 1) by adding steps `00, 09, 10`, so that the auxiliary experiment made by M runs of such a MC results in a sample of M DAGs:

```
INPUT: number of nodes n, number of
iterations N, number of DAGs M.
```

```
OUTPUT: a vector of counts.
00.Repeat M times:
01.Initialize a simple tree in which each
   node has just one parent, except the
   root node without parents;
02.Repeat the next loop N times:
03    Generate uniformly  a pair of
      distinct nodes i,j;
04    If arc(i,j) exists in the graph,
         delete the arc providing the
         graph remains connected;
05    else
06    Add the arc, provided that the
      graph  remains a DAG;
07    Otherwise keep the same state;
08.Return the current graph after N
   iterations;
09.Assign the returned DAG to an
   element of the partition;
10.Return the vector of counts after
   M iterations;
```

Proposition 4 (MC estimate of cardinalities)**.** *Given a Markov Chain algorithm providing DAGs uniformly sampled from the spaces of DAGs on a fixed set V, an estimate of cardinality $n_j, j \in \mathcal{J}$, of elements in the canonical partition $\mathcal{C}$ is:*

$$\widehat{n}_j = \frac{f(n_K)}{M} \cdot \sum_{i=1}^{M} \mathbf{I}_{\mathcal{C}_j}(z_i) \tag{3}$$

with $f(n_K)$ the total number of DAGs on a fixed set V and M the number of simulated chains. The indicating function $\mathbf{I}_{\mathcal{C}_j}(z_i)$ is equal to 1 if the structure z_i belongs to $\mathcal{C}_j$, zero otherwise.

Proof. The above algorithm generates a sample of M DAGs. Each DAG is assigned to the equivalence class in $\mathcal{C}$ which corresponds to the canonical feature $\mathcal{F}_{j(z)}$. The total number of DAGs on a fixed set of nodes V is obtained by recursion (Robinson, 1977):

$$f(n_K) = \sum_{i=1}^{n_K} (-1)^{i+1} \cdot \binom{n_K}{i} \cdot 2^{i \cdot (n_K - i)} \cdot f(n_K - i) \tag{4}$$

where $f(1) = 1, f(0) = 1$ and $n_K \geq 2$. □

2.4 Coherence, stability and revision

The elicitation of expert beliefs is not made by one straight operation. It is closer to a self-untangling adaptive procedure which increase in clarity during its dynamic. In this perspective the need of revision and elaboration of elicited values is pretty understandable and generally accepted in practice. The psychological nature of the elicitation process may lead to poorly elicited quantities, as it has been discussed in the literature (Garthwaite et al., 2005, and references therein). For this reason it is convenient to elicit more quantities than needed, that is a redundant collection of conditional odds is elicited.

Definition 6. (Coherent anchoring) Let $\widetilde{\omega}_{\mathcal{R}}$ be the collection of distinct complete minimal ensembles based on $\mathcal{R}$, that is ensembles in which at least one value among conditional odds is built on features taken from different equivalence classes. Then degree of beliefs are coherently anchored if all complete minimal ensembles provide the same distribution of subjective probability values on $\mathcal{F}$.

Elaboration of elicited quantities is performed to improve the correspondence between expert's belief and numerical assignments. Coherent anchoring leads to the definition of a probability measure on the algebra of features $\mathcal{A}(\mathcal{F})$. Subjective probability values for marginal canonical features may be compared to actual expert beliefs about the same joint statements for the unknown structure.

Definition 7 (Reduced reference set)**.** Let $\mathcal{R}$ be a set of reference features. A reduced reference set $\mathcal{R}_r$ is a proper subset of $\mathcal{R}$.

Proposition 5 (Stability)**.** *Let $\widetilde{\mathcal{R}}$ be the collection of all reduced reference sets obtained from a given reference set: $\widetilde{\mathcal{R}} = \{\mathcal{R}_r : \mathcal{R}_r \subset \mathcal{R}.\}$. Let $\widetilde{\omega}_{\mathcal{R}_r}$ be a collection of distinct complete minimal ensembles, as in Definition 6, based on $\mathcal{R}_r$ and the preference relation $\mathcal{U}_r$. Then elicited degree of beliefs are stable under reduction $\mathcal{R}_r$ if:*

$$P\left[\mathcal{F}_j \mid \mathcal{U}_r, \mathcal{R}_r, \xi\right] = P\left[\bigvee_{s \in S} \mathcal{F}_s \mid \mathcal{U}, \mathcal{R}, \xi\right], \tag{5}$$

where S is the collection of index values denoting canonical features based on $\mathcal{R}$ which appear in the disjunction producing the canonical feature $\mathcal{F}_j$ based on $\mathcal{R}_r$. The elicited degree of beliefs are stable if they are stable for all reductions in $\widetilde{\mathcal{R}}$.

Proof. If one or more features are removed from a reference set then the canonical basis of elicitation partially collapses to one of larger granularity. The associated algebra is given by unions of elements taken from the starting algebra. □

Full stability may be heavy to check and a useful compromise is to limit the number of reductions, for example to the collection of all one-feature reference sets. The revision of elicited beliefs is mandatory if stability or coherent anchoring are violated for some reductions.

2.5 A case study in breast cancer

Classical biomarkers in breast cancer studies include progesterone receptors (PR), oestrogen receptors (ER), age (AG), menopausal status (MS), number of positive lymph nodes (PL). Variables of interest for patients are tumor grade (TG) and tumor size (TS). The reference set $\mathcal{R}$ contains the propositions below:
$\mathcal{P}_1$ ='Nodes AG, MS precede all other nodes';
$\mathcal{P}_2$ ='TS, TG, NL follow all other nodes';
$\mathcal{P}_3$ ='The parent set is made by three or less nodes for each node in V';
$\mathcal{P}_4$ ='ER is independent on AG given MS'.

An important particularization of the general elicitation scheme is obtained by a preference relation $\mathcal{U}$ which sets the order over canonical features according to the the number of true propositions making each canonical feature. The canonical feature $\neg\mathcal{P}_1 \wedge \neg\mathcal{P}_2 \wedge \neg\mathcal{P}_3 \wedge \neg\mathcal{P}_4$ precedes all the other canonical features, while $\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \mathcal{P}_3 \wedge \mathcal{P}_4$ succeeds to all the other canonical features. It follows that the first and last equivalence classes are: $\mathcal{E}_0 = \{\neg\mathcal{P}_1 \wedge \neg\mathcal{P}_2 \wedge \neg\mathcal{P}_3 \wedge \neg\mathcal{P}_4\}$ and $\mathcal{E}_4 = \{\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \mathcal{P}_3 \wedge \mathcal{P}_4\}$. Four canonical features have just one proposition true and they define the equivalence class $\mathcal{E}_1 = \{\neg\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \mathcal{P}_3 \wedge \mathcal{P}_4, \mathcal{P}_1 \wedge \neg\mathcal{P}_2 \wedge \mathcal{P}_3 \wedge \mathcal{P}_4, \mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \neg\mathcal{P}_3 \wedge \mathcal{P}_4, \mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \mathcal{P}_3 \wedge \neg\mathcal{P}_4\}$ which follows $\mathcal{E}_0$. Equivalence classes $\mathcal{E}_2$ and $\mathcal{E}_3$ are defined in a similar way. The cardinality of an equivalence class $\mathcal{E}_i$ in this case study is equal to $\mid \mathcal{E}_i \mid = \binom{n_f}{i}$, where n_f is the total number of propositions and i the number of true propositions for each canonical feature in the equivalence class $\mathcal{E}_i$: $\mid \mathcal{E}_0 \mid = 1$, $\mid \mathcal{E}_1 \mid = 4$, $\mid \mathcal{E}_2 \mid = 6$, $\mid \mathcal{E}_3 \mid = 4$, $\mid \mathcal{E}_4 \mid = 1$. In this particular preference relation the total number of equivalence classes within the partition $\mathcal{E}$ is $n_e = n_f + 1 = 5$.

In the elicitation, three distinct complete minimal ensembles are considered, say $\widetilde{\omega}_{\mathcal{R}} = \{\widetilde{\omega}_{\mathcal{R},1}, \widetilde{\omega}_{\mathcal{R},2}, \widetilde{\omega}_{\mathcal{R},3}\}$. The first ensemble is: $\widetilde{\omega}_{\mathcal{R},1} = \{\omega_{1,0} = 2.0, \omega_{2,0} = 3.0, \omega_{3,0} = 4.0, \omega_{4,0} = 5.0\}$, with indices $i = 0, 1, 2, 3, 4$ denoting any canonical feature belonging to equivalence class $\mathcal{E}_i$. The second and third ensembles are: $\widetilde{\omega}_{\mathcal{R},2} = \{\omega_{1,0} = 1, \omega_{2,1} = \frac{3}{2}, \omega_{3,2} = \frac{4}{3}, \omega_{4,3} = \frac{5}{4}\}$, $\widetilde{\omega}_{\mathcal{R},3} = \{\omega_{0,4} = \frac{1}{5}, \omega_{1,3} = \frac{1}{2}, \omega_{2,4} = \frac{3}{5}, \omega_{3,2} = \frac{4}{3}\}$. The anchoring is coherent because the three ensembles provide the same probability values: $P[\mathcal{F}_{[0]} \mid \mathcal{U}, \xi] = \frac{1}{48}$, $P[\mathcal{F}_{[1]} \mid \mathcal{U}, \xi] = \frac{1}{24}$, $P[\mathcal{F}_{[2]} \mid \mathcal{U}, \xi] = \frac{1}{16}$, $P[\mathcal{F}_{[3]} \mid \mathcal{U}, \xi] = \frac{1}{12}$, $P[\mathcal{F}_{[4]} \mid \mathcal{U}, \xi] = \frac{5}{48}$.

The algorithm described in Section (2.3) run with parameters $M = 10000$ and $N = 294$. In (Ide et al., 2002) the authors motivated the choice of $N = K^2 * 6$ through empirically findings. We replicated the above simulation for one hundred times to assess the variability of point estimates of the fractions of DAGs in $\mathcal{C}_j, j \in \mathcal{J}$ (full results not shown): minimum and maximum standard deviations of $\hat{n}_j$ observed for the $2^4 = 16$ elements of the canonical partition $\mathcal{C}$ are, respectively, 0.000475 and 0.004797 in 100 replicated simulations. The total number of DAGs for seven nodes is $f(7) = 1'138'779'265$ (equation 4).

The stability of elicited values has been examined limited to four reduced reference sets: $\widetilde{\mathcal{R}} = \{\mathcal{R}_i : i = 1, 2, 3, 4\}$, so that the reduced reference set $\mathcal{R}_i = \{P_i\}$ contains just one proposition. An explicit expression exploiting the already introduced index set $\mathcal{J}$ is easily obtained for such reductions, for example: $P[\mathcal{F}_{[1]} \mid \mathcal{U}_1, \mathcal{R}_1, \xi] = \sum_{s \in S} P[\mathcal{F}_s \mid \mathcal{R}, \mathcal{U}, \widetilde{\omega}_R, \xi]$, where $S = \{(1, \bar{2}, \bar{3}, \bar{4}), \ldots, (1, 2, 3, 4)\}$. In this way we obtained four marginal probability val-

ues of each proposition $\mathcal{P}_i, i = 1, 2, 3, 4$ and they are all equal to $\frac{7}{12}$. The expert did not reject the above values which follow from the unrestricted elicitation, so the revision did not take place (see the discussion).

3 Discussion

The generality of the approach described in this work is mainly due to the use of propositions describing network features. Nevertheless the usefulness also depends on the amount of work left to the expert in actual problem domains. The good scaling of the proposed elicitation with an increasing number of nodes rests on a number of propositions which is far smaller than the number of DAGs. Moreover structures counting is left to simulation, and reasonable estimates in equation (3) are obtained by sampling a number of DAGs M which is much smaller than the total number of DAGS $f(n_K)$. Even for an increasing number of propositions and very large spaces of structures the computation may remain feasible if the order relation $\mathcal{U}$ induces large equivalence classes of canonical features. Nevertheless, the overall computational burden partially depends on the nature of features. For example, features local to Markov blankets are quickly checked, while features involving the consideration of the whole structure are computationally heavy to assess.

Particularized instances of our approach may serve as starting elicitation from which semi-automatic prior distributions may be quickly obtained. The preference relation $\mathcal{U}$ discussed in the case study induces a partition in which $\mathcal{E}_i$ collects all canonical features made by i true propositions. A computationally efficient assignment of probability values to canonical features is obtained by eliciting a value $0 < k < 1$ and by setting $P[\mathcal{F}_{[i]} \mid \mathcal{U}, \xi] \propto k^{(n_f - i)}$, so that for two features $\mathcal{F}_{[i']}$ and $\mathcal{F}_{[i'']}$ satisfying $i' - i'' = \delta$ we have $\omega_{i',i''} = k^{-\delta}$. Reconciliation of incoherent anchoring may be automatically performed by defining an equally weighted mixture of probability distributions obtained from different ensembles. This reconciliation scheme might be an automatic first step towards a detailed revision and it could be useful in problem domains with a large number of features and weak prior information.

Simpler elicitation schemes may work in practice, and in these setups our formalization may be useful in obtaining the elicitation bias, for example, due to the assumption of a-priori independence among propositions. Sharp prior information has been coded as structural restrictions in (de Campos and Castellano, 2007), but it may be as well coded using canonical features and degree of beliefs very close to 0 or to 1, so that management of restrictions and potential overstatement of beliefs are avoided. A comparison of computational burden and of flexibility between the two approaches in similar case studies is a theme for future research. The elicitation based on network features and conditional odds is more demanding than the approach in (Castelo and Siebes, 2000), where edges are units of elicitation that are combined up to a full prior for a BN in a quite implicit way. The use of oriented graphs and the distribution of the extra-DAGs amount of weight lead to a prior distribution for BNs, but in large sized problem domains the resulting prior distribution may be difficult to submit to further inspection as regards non-local properties. Levels of information including full structures, correlation or causation among nodes, temporal order (O'Donnell et al., 2006) may be also captured through features, while the implications due to the use of a uniform distribution on the space of Totally Ordered Models (O'Donnell et al., 2006, TOM) has still to be investigated. Finally, at the best of our knowledge the approach described in this work seems the only one to reach full generality in considering global network features, therefore numerical explorations of its performances under common learners, like greedy search, deserve research efforts.

A key step of our scheme is the auxiliary Monte Carlo experiment which provides the cardinality of equivalence classes when straight counting DAGs is too heavy. The algorithm discussed in (Ide et al., 2004) may be used to count DAGs in restricted spaces, for example for a sharply believed feature like "no more then

three nodes in each parent set" with a probability equal to 1. The counting problem has been also considered by (Peña, 2007), who provides MCMC algorithms to approximately calculate the ratio of DAG models to DAGs up to 20 nodes and the fraction of chain graph models that are neither DAG models nor DAG models up to 13 nodes. The extension of the approach to elicitation described in this work for more general classes of graphs deserves attention in future work.

4 Conclusions

In this paper we proposed a formal procedure to elicit expert beliefs on the structure of a Bayesian network by means of propositions which capture relevant features. While a detailed elicitation may be overwhelming for the expert in large problem domains, particularizations of the general approach offer automatic completion and limited expert efforts.

Further work on elicitation is needed both on theoretical and applied sides. An inferential engine could suggest sure-false and sure-true canonical features given a reference set. Moreover a graphical user interface could make elicitation and revision easier to perform for applied scientists. It is has been found that different propositions embedding the same meaning may lead to different elicited values. Finally, human cognitive peculiarities related to the elicitation of beliefs on plausible structures for a BN are still largely unexplored.

Acknowledgments

This work is funded by Italian MIUR (PRIN). The author thanks anonymous referees for their comments.

References

Wray Buntine. 1991. Theory of Refinement on Bayesian Networks, In *Proceedings of 7th Conference on Uncertainty in Artificial Intelligence*, pages 52–60.

Roberto Castelo and Arno Siebes. 2000. Priors on network structures. Biasing the search for Bayesian networks, *International Journal of Approximate Reasoning*, 24:39–57.

Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen and David J. Spiegelhalter. 1999. *Probabilistic Networks and expert systems*, Springer Verlag.

Luis M. de Campos and Javier G. Castellano. 2007. Bayesian network learning algorithms using structural restrictions, *International Journal of Approximate Reasoning*, 45:233–254.

Paul H. Garthwaite, Joseph B. Kadane and Antony O'Hagan. 2005. Statistical methods for eliciting probability distributions, *Journal of the American Statistical Association*, 100:680–701.

David Heckerman, Dan Geiger, David M. Chickering. 1995. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data, *Machine Learning*, 20:197–243.

Jaime S. Ide and Fabio Gagliardi Cozman. 2002. Random generation of Bayesian Networks. In *Proceedings of the Brazilian Symposium on Artificial Intelligence*, Brazil, pages 366–375.

Jaime S. Ide, Fabio Gagliardi Cozman and Fabio T. Ramos. 2004. Generating Random Bayesian Networks with Constraints on Induced Width. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04)*, IOS Press, Amsterdam, pages 323-327.

Massimiliano Mascherini and Federico M. Stefanini. 2007. Using weak prior information on structures to learn Bayesian Networks. In *B. Apolloni et al. (Eds.): KES 2007/WIRN 2007,Part I, LNAI 4692*, Springer Verlag, Berlin, pages 413-420.

Rodney T. O'Donnell, Ann E. Nicholson, Bin Han, Kevin B. Korb, Jahangir M. Alam and Lucas R. Hope. 2006. Causal discovery with prior information. In *A: Sattar and B. H. Kang (Eds): AI 2006, LNAI 4304*, Springer Verlag, Berlin, pages 1162-1167.

Jose M. Peña. 2007. Approximate counting of Graphical Models via MCMC, In *Proceedings of 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, pages 352–359.

Robert W. Robinson. 1997. Counting unlabeled acyclic digraphs. In *Combinatorial Mathematics V* (C.H.C. Little, ed.) Springer Lectures Notes in Mathematics 622, pages 28–43.

A Geometric Approach to Learning BN Structures

M. Studený and J. Vomlel
Institute of Information Theory and Automation of the ASCR
Prague, CZ 18208, Czech Republic

Abstract

We recall the basic idea of an algebraic approach to learning a Bayesian network (BN) structure, namely to represent every BN structure by a certain (uniquely determined) vector, called *standard imset*. The main result of the paper is that the set of standard imsets is the set of vertices (= extreme points) of a certain polytope. Motivated by the geometric view, we introduce the concept of the *geometric neighborhood* for standard imsets, and, consequently, for BN structures. To illustrate this concept by an example, we describe the geometric neighborhood in the case of three variables and show it differs from the *inclusion neighborhood*, which was introduced earlier in connection with the GES algorithm. This leads to an example of the failure of the GES algorithm if data are not "generated" from a perfectly Markovian distribution. The point is that one can avoid this failure if the greedy search technique is based on the geometric neighborhood instead.

1 Introduction

The motivation for this theoretical paper is learning a Bayesian network (BN) structure from data by the method of maximization of a quality criterion (= the score and search method). By a *quality criterion*, also named a *score metric* by other authors, we mean a real function $\mathcal{Q}$ of the BN structure, usually represented by a graph G, and of the database D. The value $\mathcal{Q}(G, D)$ "evaluates" how the BN structure given by G fits the database D.

An important related question is how to represent a BN structure in the memory of a computer. Formerly, each BN structure was represented by an arbitrary acyclic directed graph defining it, which led to the non-uniqueness in its description. Later, researchers calling for methodological simplification came up with the idea to represent every BN structure with a unique representative. The most popular graphical representative is the *essential graph*. It is a chain graph describing shared features of acyclic directed graphs defining the BN structure. The adjective "essential" was proposed by Andersson, Madigan and Perlman (1997), who gave a graphical characterization of essential graphs.

Since direct maximization of a quality criterion $\mathcal{Q}$ seems, at first sight, to be infeasible, various *local search methods* have been proposed. The basic idea is that one introduces a neighborhood relation between BN structure representatives, also named *neighborhood structure* by some authors (Bouckaert, 1995). Then one is trying to find a local maximum with respect to the chosen neighborhood structure. This is an algorithmically simpler task because one can utilize various greedy search techniques for this purpose. On the other hand, the algorithm can get stuck in a local maximum and fail to find the global maximum. A typical example of these techniques is *greedy equivalence search* (GES) algorithm proposed by Meek (1997). The neighborhood structure utilized in this algorithm is the *inclusion neighborhood*, which comes from the conditional independence interpretation of BN structures. Chickering (2002) proposed a modification of the GES algorithm, in which he used the essential graphs as (unique) BN structure representatives.

There are two important technical requirements on a quality criterion $\mathcal{Q}$ brought in connection with the local search methods, namely to make them computationally feasible. One

of them is that $\mathcal{Q}$ should be *score equivalent* (Bouckaert, 1995), which means it ascribes the same value to equivalent graphs. The other requirement is that $\mathcal{Q}$ should be *decomposable* (Chickering, 2002), which means that $\mathcal{Q}(G, D)$ decomposes into contributions which correspond to the factors in the factorization according to the graph G.

The basic idea of an algebraic approach to learning BN structures, presented in Chapter 8 of (Studený, 2005), is to represent both the BN structure and the database with real vectors. More specifically, an algebraic representative of the BN structure defined by an acyclic directed graph G is a certain integer-valued vector u_G, called the *standard imset* (for G). It is also a unique BN structure representative because $u_G = u_H$ for equivalent graphs G and H. Another boon of standard imsets is that one can read practically immediately from the differential imset $u_G - u_H$ whether the BN structures defined by G and H are neighbors in the sense of inclusion neighborhood. However, the crucial point is that every score equivalent and decomposable criterion $\mathcal{Q}$ is an affine function (= linear function plus a constant) of the standard imset. More specifically, it is shown in § 8.4.2 of (Studený, 2005) that one has

$$\mathcal{Q}(G, D) \,=\, s_D^{\mathcal{Q}} - \langle t_D^{\mathcal{Q}}, u_G \rangle \,,$$

where $s_D^{\mathcal{Q}}$ is a real number, $t_D^{\mathcal{Q}}$ a vector of the same dimension as the standard imset u_G (they both depend solely on the database D and the criterion $\mathcal{Q}$) and $\langle *, * \rangle$ denotes the scalar product. The vector $t_D^{\mathcal{Q}}$ is named the *data vector* (relative to $\mathcal{Q}$).

We believe the above-mentioned result paves the way for future application of efficient linear programming methods in the area of learning BN structures. This paper is a further step in this direction: its aim is to enrich the algebraic approach by a geometric view. One can imagine the set of all standard imsets over a fixed set of variables N as the set of points in the respective Euclidean space. The main result of the paper is that it is the set of vertices (= extreme points) of a certain polytope. One consequence of this result is as follows: since every "reasonable" quality criterion $\mathcal{Q}$ can be viewed as (the restriction of) an affine function on the respective Euclidean space, the task to maximize $\mathcal{Q}$ over standard imsets is equivalent to the task of maximizing an affine function (= the extension) over the above-mentioned polytope.

Now, a well-known classic result on convex sets in the Euclidean space, Weyl-Minkowski theorem, says that a polytope can equivalently be introduced as a bounded polyhedron. Thus, once one succeeds in describing the above-mentioned polytope in the form of a (bounded) polyhedron, one gets a classic task of linear programming, namely to find an extremal value of a linear function over a polyhedron. There are efficient methods, like the *simplex method*, to tackle this problem (Schrijver, 1986). To illustrate the idea we describe the above-mentioned (standard imset) polytope in the form of a bounded polyhedron in the case $|N| = 3$ in the paper and give a web reference for $|N| = 4$.

However, because it is not clear at this moment how to find the "polyhedral" description of the polytope for arbitrary $|N|$, we propose an alternative approach in this paper. The basic idea is to introduce the concept of *geometric neighborhood* for standard imsets, and, therefore, for BN structures as well. The standard imsets u_G and u_H will be regarded as (geometric) neighbors if the line-segment connecting them is a face of the polytope (= the edge of the polytope in the geometric sense). The motivation is as follows: one of possible interpretations of the simplex method is that it is a kind of "greedy search" method in which one moves between vertices (of the polyhedron) along the edges - see § 11.1 of (Schrijver, 1986). Thus, provided one succeeds at characterizing the geometric neighborhood, one can possibly use greedy search techniques to find the global maximum of $\mathcal{Q}$ over the polytope, and, therefore, over the set of standard imsets. To illustrate the concept of geometric neighborhood we characterize it for 3 variables in the paper and give a web reference to the characterization in the case of 4 variables.

The finding is that the inclusion neighbor-

hood and geometric neighborhood differ already in the case of 3 variables. This observation has a simple but notable consequence: the GES algorithm, which is based on the inclusion neighborhood, may fail to find the global maximum of $\mathcal{Q}$. We give such an example and claim that this is an inevitable defect of the inclusion neighborhood, which may occur whenever the data faithfulness is not guaranteed. In our view, the data faithfulness relative to a perfectly Markovian distribution is a very strong unrealistic assumption except for the case of artificially generated data.

2 Basic Concepts

In this section we recall basic definitions and results concerning learning BN structures.

2.1 BN Structures

One of the possible definitions of a (discrete) *Bayesian network* is that it is a pair (G, P), where G is an acyclic directed graph over a (non-empty finite) set of nodes (= variables) N and P a discrete probability distribution over N that (recursively) factorizes according to G (Neapolitan 2004). A well-known fact is that P factorizes according to G iff it is Markovian with respect to G, which means it satisfies the conditional independence restrictions determined by the graph G through the corresponding (directed) separation criterion (Pearl, 1988; Lauritzen, 1996). Having fixed (non-empty finite) sample spaces X_i for variables $i \in N$, the respective (BN) *statistical model* is the class of all probability distributions P on $\mathsf{X}_N \equiv \prod_{i\in N} \mathsf{X}_i$ that factorize according to G. To name the shared features of distributions in this class one can use the phrase "*BN structure*". Of course, the structure is determined by the graph G, but it may happen that two different graphs over N describe the same structure.

2.1.1 Equivalence of graphs

Two acyclic directed graphs over N will be named *Markov equivalent* if they define the same BN statistical model. If $|\mathsf{X}_i| \geq 2$ for every $i \in N$, then this is equivalent to the condition they are *independence equivalent*, which means they determine the same collection of conditional independence restrictions – cf. § 2.2 in (Neapolitan, 2004). Both Frydenberg (1990), and Verma and Pearl (1991) gave classic graphical characterization of independence equivalence: two acyclic directed graphs G and H over N are independence equivalent iff they have the same underlying undirected graph and *immoralities*, i.e. induced subgraphs of the form $a \to c \leftarrow b$, where $[a, b]$ is not an edge in the graph.

2.1.2 Learning BN structure

The goal of (structural) learning is to determine the BN structure on the basis of data. These are assumed to have the form of a *complete database* $D : x^1, \ldots, x^d$ of the length $d \geq 1$, that is, of a sequence of elements of X_N. Provided the sample spaces X_i with $|\mathsf{X}_i| \geq 2$ for $i \in N$ are fixed, let $\mathsf{DATA}(N, d)$ denote the collection of all databases over N of the length d. Moreover, let $\mathsf{DAGS}(N)$ denote the collection of all acyclic directed graphs over N. Then we take a real function $\mathcal{Q}$ on $\mathsf{DAGS}(N) \times \mathsf{DATA}(N, d)$ for a *quality criterion*. The value $\mathcal{Q}(G, D)$ should reflect how the statistical model determined by G is suitable for explaining the (occurrence of the database) D. The learning procedure based on $\mathcal{Q}$ then consists in maximization of the function $G \mapsto \mathcal{Q}(G, D)$ over $G \in \mathsf{DAGS}(N)$ if the database $D \in \mathsf{DATA}(N, d)$, $d \geq 1$ is given.

A classic example is Jeffreys-Schwarz *Bayesian information criterion* (BIC), defined as the maximum of the likelihood minus a penalty term, which is a multiple of the number of free parameters in the statistical model (Schwarz, 1978). To give a direct formula for BIC (in this case) we need a notational convention. Given $i \in N$, let $r(i)$ denote the cardinality $|X_i|$, $pa_G(i) \equiv \{j \in N; j \to i\}$ the set of *parents* of i in $G \in \mathsf{DAGS}(N)$, and $q(i, G) \equiv |\prod_{j\in pa_G(i)} \mathsf{X}_j|$ the number of parent configurations for i (in G). Provided $i \in N$ is fixed, the letter k will serve as a generic symbol for (the code of) an element of X_i (= a node configuration) while j for (the code of) a parent configuration. Given a database D of the length $d \geq 1$ let d_{ijk} denote the

number of occurences in D of the (marginal) parent-node configuration encoded by j and k; put $d_{ij} = \sum_{k=1}^{r(i)} d_{ijk}$. Here is the formula - see Corollary 8.2 in (Studený, 2005):

$$\begin{aligned}\mathsf{BIC}\,(G,D) \;&=\; \sum_{i\in N}\sum_{j=1}^{q(i,G)}\sum_{k=1}^{r(i)} d_{ijk}\cdot\ln\frac{d_{ijk}}{d_{ij}} \\ &\quad -\frac{\ln d}{2}\cdot\sum_{i\in N} q(i,G)\cdot[r(i)-1]\,.\end{aligned}$$

In this brief overview we omit the question of statistical consistency of quality criteria; we refer the reader to the literature on this topic (Chickering, 2002; Neapolitan, 2004). A quality criterion $\mathcal{Q}$ will be named *score equivalent* if, for every $D \in \mathsf{DATA}(N,d)$, $d \geq 1$,

$$\mathcal{Q}(G,D) = \mathcal{Q}(H,D) \quad \text{if } G,H \in \mathsf{DAGS}(N)$$

are independence equivalent. Moreover, $\mathcal{Q}$ will be called *decomposable* if there exists a collection of functions $q_{i|B} : \mathsf{DATA}(\{i\}\cup B, d) \to \mathbb{R}$ where $i \in N$, $B \subseteq N\setminus\{i\}$, $d \geq 1$ such that, for every $G \in \mathsf{DAGS}(N)$, $D \in \mathsf{DATA}(N,d)$ one has

$$\mathcal{Q}(G,D) = \sum_{i\in N} q_{i|pa_G(i)}(D_{\{i\}\cup pa_G(i)})\,,$$

where $D_A : x_A^1,\ldots,x_A^d$ denotes the projection of D to the marginal space $\mathsf{X}_A \equiv \prod_{i\in A}\mathsf{X}_i$ for $\emptyset \neq A \subseteq N$.

2.1.3 Inclusion neighborhood

The basic idea of local search methods for the maximization of a quality criterion (= score and search methods) has already been explained in the Introduction. Now, we define the inclusion neighborhood formally. Given $G \in \mathsf{DAGS}(N)$, let $\mathcal{I}(G)$ denote the collection of conditional independence restrictions determined by G. Given $G,H \in \mathsf{DAGS}(N)$, if $\mathcal{I}(H) \subset \mathcal{I}(G)$,[1] but there is no $F \in \mathsf{DAGS}(N)$ with $\mathcal{I}(H) \subset \mathcal{I}(F) \subset \mathcal{I}(G)$, then we say H and G are *inclusion neighbors*. Of course, this terminology can be extended to the corresponding BN structures and their representatives.

[1]Here, $\mathcal{I} \subset \mathcal{J}$ denotes strict inclusion, that is, $\mathcal{I} \subseteq \mathcal{J}$ but $\mathcal{I} \neq \mathcal{J}$.

Note that one can test graphically whether $G,H \in \mathsf{DAGS}(N)$ are inclusion neighbors; this follows from transformational characterization of inclusion $\mathcal{I}(H) \subseteq \mathcal{I}(G)$ provided by Chickering (2002).

2.1.4 Essential graph

Given an (independence) equivalence class $\mathcal{G}$ of acyclic directed graphs over N, the respective *essential graph* G^* is a hybrid graph (= a graph with both directed and undirected edges) defined as follows:

- $a \to b$ in G^* if $a \to b$ in every $G \in \mathcal{G}$,
- $a - b$ in G^* if there are $G,H \in \mathcal{G}$ such that $a \to b$ in H and $a \leftarrow b$ in G.

It is always a chain graph (= acyclic hybrid graph); this follows from graphical characterization of (graphs that are) essential graphs by Andersson, Madigan and Perlman (1997). Chickering (2002) used essential graphs as unique graphical BN structure representatives in his version of the GES algorithm.

2.2 Standard Imset

By an *imset* u over N will be meant an integer-valued function on the power set of N, that is, on $\mathcal{P}(N) \equiv \{A;\, A \subseteq N\}$. We will regard it as a vector whose components are integers and are indexed by subsets of N. Actually, any real function $m : \mathcal{P}(N) \to \mathbb{R}$ will be interpreted as a (real) vector in the same way, that is, identified with an element of $\mathbb{R}^{\mathcal{P}(N)}$. The symbol $\langle m,u\rangle$ will denote the scalar product of two vectors of this type:

$$\langle m,u\rangle \equiv \sum_{A\subseteq N} m(A)\cdot u(A)\,.$$

To write formulas for imsets we introduce the following notational convention. Given $A \subseteq N$, the symbol δ_A will denote a special imset:

$$\delta_A(B) = \begin{cases} 1 & \text{if } B = A, \\ 0 & \text{if } B \neq A, \end{cases} \qquad \text{for } \; B \subseteq N.$$

By an *elementary imset* is meant an imset

$$u_{\langle a,b|C\rangle} \;=\; \delta_{\{a,b\}\cup C} + \delta_C - \delta_{\{a\}\cup C} - \delta_{\{b\}\cup C}\,,$$

where $C \subseteq N$ and $a, b \in N \setminus C$ are distinct. In our algebraic framework it encodes an elementary conditional independence statement $a \perp\!\!\!\perp b \,|\, C$.

Given $G \in \mathsf{DAGS}(N)$, the *standard imset* for G, denoted by u_G, is given by the formula

$$u_G \;=\; \delta_N - \delta_\emptyset + \sum_{i \in N} \{\, \delta_{pa_G(i)} - \delta_{\{i\} \cup pa_G(i)} \,\}. \quad (1)$$

It follows from (1) that u_G has at most $2 \cdot |N|$ non-zero values. Thus, one can keep only its non-zero values, which means the memory demands for representing standard imsets are polynomial in the number of variables.

It was shown as Corollary 7.1 in (Studený, 2005) that, given $G, H \in \mathsf{DAGS}(N)$, one has $u_G = u_H$ iff they are independence equivalent. Moreover, Corollary 8.4 in Studený (2005) says that $G, H \in \mathsf{DAGS}(N)$ are inclusion neighbors iff either $u_G - u_H$ or $u_H - u_G$ is an elementary imset. Finally, Lemmas 8.3 and 8.7 in (Studený, 2005) together claim that every score equivalent and decomposable criterion $\mathcal{Q}$ necessarily has the form:

$$\mathcal{Q}(G, D) \;=\; s^{\mathcal{Q}}_D - \langle t^{\mathcal{Q}}_D, u_G \rangle \quad (2)$$

for $G \in \mathsf{DAGS}(N)$, $D \in \mathsf{DATA}(N, d)$, $d \geq 1$ where the constant $s^{\mathcal{Q}}_D \in \mathbb{R}$ and the (data) vector $t^{\mathcal{Q}}_D : \mathcal{P}(N) \to \mathbb{R}$ do not depend on G.

The reader can object that the dimension of $t^{\mathcal{Q}}_D$ grows exponentially with $|N|$, making the method unfeasible for many "real-world" problems. However, since $2^{|N|} \leq |\mathsf{X}_N|$, the representation of a database D in the form of a data vector may appear to be even more effective than (one of the traditional ways) in the form of a contingency table! Another point is that to compute $\langle t^{\mathcal{Q}}_D, u_G \rangle$ one only needs at most $2 \cdot |N|$ values of the data vector. In brief, we believe that whenever one is able to represent the database in the memory of a computer then one should be able to take care of the data vector as well.

3 Some Geometric Concepts

In this section we recall well-known concepts and facts from the theory of convex polytopes (Schrijver, 1986).

3.1 Polytopes and Polyhedrons

These sets are special subsets of the Euclidean space $\mathbb{R}^K$, where K is a non-empty finite set. The points in this space are vectors $\boldsymbol{v} = [v_i]_{i \in K}$. Given $\boldsymbol{x}, \boldsymbol{v} \in \mathbb{R}^K$ their scalar product is $\langle \boldsymbol{v}, \boldsymbol{x} \rangle = \sum_{i \in K} v_i \cdot x_i$.

A *polytope* in $\mathbb{R}^K$ is the convex hull of a finite set of points in $\mathbb{R}^K$; if the set consists of points in $\mathbb{Q}^K$, the polytope is *rational*. It is straightforward that the smallest set of points whose convex hull is a polytope P is the set of its *vertices* ($\equiv$ *extreme points*), that is, of those points in P which cannot be written as convex combinations of the other points in P. In particular, the set of vertices of P is finite. The *dimension* $\dim(\mathsf{P})$ of $\mathsf{P} \subseteq \mathbb{R}^K$ is the dimension of its affine hull $\mathrm{aff}(\mathsf{P})$, which is the collection of affine combinations $\sum_{\boldsymbol{v} \in \mathsf{R}} \lambda_{\boldsymbol{v}} \cdot \boldsymbol{v}$, where $\emptyset \neq \mathsf{R} \subseteq \mathsf{P}$ is finite and $\lambda_{\boldsymbol{v}} \in \mathbb{R}$, $\sum_{\boldsymbol{v} \in \mathsf{R}} \lambda_{\boldsymbol{v}} = 1$.[2] A polytope is *full-dimensional* if $\dim(\mathsf{P}) = |K|$.

An *affine half-space* in $\mathbb{R}^K$ is the set

$$H^+ = \{\boldsymbol{x} \in \mathbb{R}^K;\, \langle \boldsymbol{v}, \boldsymbol{x} \rangle \leq \alpha\}\,,$$

where $0 \neq \boldsymbol{v} \in \mathbb{R}^K$ and $\alpha \in \mathbb{R}$. A *polyhedron* is the intersection of finitely many affine half-spaces. It is *bounded* if it does not contain a ray $\{\boldsymbol{x} + \alpha \cdot \boldsymbol{w};\, \alpha \geq 0\}$ for any $\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^K$, $\boldsymbol{w} \neq 0$.

A well-known classic, but non-trivial, result is that $\mathsf{P} \subseteq \mathbb{R}^K$ is a polytope iff it is a bounded polyhedron – see Corollary 7.1.c in (Schrijver, 1986). A further important observation is that if P is a full-dimensional polytope then its *irredundant description* in the form of a polyhedron[3] is unique – see claim (17) on page 102 of (Schrijver, 1986).

Finally, there are software packages that allow one, on the basis of the list of vertices of a rational polytope P, to compute all inequalities defining an irredundant polyhedral description of P, e.g. (Franz, 2006).

[2] There is a unique linear subspace $\mathsf{L} \subseteq \mathbb{R}^K$ such that $\mathrm{aff}(\mathsf{P}) = \boldsymbol{w} + \mathsf{L}$ for some $\boldsymbol{w} \in \mathbb{R}^K$. The dimension of $\mathrm{aff}(\mathsf{P})$ is defined as the dimension of L.

[3] By this is meant the intersection of such a collection of half-spaces in which no half-space can be dropped without changing the polyhedron.

4 Main Result

In this section we give the main result and illustrate it in an example with three variables. Let S denote the set of standard imsets over N:

$$S \equiv \{u_G;\, G \in \mathsf{DAGS}(N)\} \subseteq \mathbb{R}^{\mathcal{P}(N)}\,.^4$$

Theorem 1. *The set S of standard imsets over N is the set of vertices of a rational polytope $\mathsf{P} \subseteq \mathbb{R}^{\mathcal{P}(N)}$. The dimension of the polytope is $2^{|N|} - |N| - 1$.*

Because of a limited scope for this paper we skip the proof, which can be found in (Studený and Vomlel, 2008).

Example Let us describe the situation in the case of three variables. Then one has 11 standard imsets and they break into 5 types (= permutation equivalence classes). They can also be classified by the number of edges in the corresponding essential graph. (c.f. Figure 1 below)

- The zero imset corresponds to the complete (undirected) essential graph.
- Six elementary imsets break into two types, namely $u_{\langle a,b|\emptyset\rangle}$ and $u_{\langle a,b|c\rangle}$; the essential graphs are $a \to c \leftarrow b$ and $a - c - b$.
- Three "semi-elementary" imsets of the form $u_{\langle a,bc|\emptyset\rangle} \equiv \delta_{abc} + \delta_\emptyset - \delta_a - \delta_{bc}$ define one type; the essential graphs have just one undirected edge.
- The imset $\delta_N - \sum_{i\in N} \delta_i + 2\cdot\delta_\emptyset$ corresponds to the empty essential graph.

By the theorem above, the dimension of the polytope generated by these 11 imsets is 4. To get its irredundant description in the form of a polyhedron it is suitable to have it embedded (as a full-dimensional polytope) in a 4-dimensional space. To this end, we decided to identify every standard imset over N with its restriction to $\mathcal{K} \equiv \{A \subseteq N;\, |A| \geq 2\}$. Then we used the computer package Convex (Franz, 2006) to get all 13 polyhedron-defining inequalities. They break into 7 types and can be classified as follows:

- Five inequalities hold with equality for the zero imset. They break into 3 types: $0 \leq 2\cdot\delta_{abc} + \delta_{ab} + \delta_{ac} + \delta_{bc}$, $0 \leq \delta_{abc} + \delta_{ab}$ and $0 \leq \delta_{abc}$.
- Eight inequalities achieve equality for the imset corresponding to the empty graph. They break into 4 types, namely $\delta_{abc} \leq 1$, $\delta_{abc} + \delta_{ab} \leq 1$, $\delta_{abc} + \delta_{ab} + \delta_{ac} \leq 1$ and $\delta_{abc} + \delta_{ab} + \delta_{ac} + \delta_{bc} \leq 1$.

We also made analogous computation in the case $|N| = 4$. In this case one has 185 standard imsets breaking into 20 types. The dimension of the polytope is 11. The number of corresponding polyhedron-defining inequalities is 154 – see vertex-facet table in (Vomlel and Studený, 2008).

Thus, in the case of three and four variables, the polyhedral description of the polytope P was found. In particular, the task to maximize a (score equivalent and decomposable) quality criterion $\mathcal{Q}$ is, by (2), equivalent to a standard linear programming problem, namely to minimize a linear function $u \mapsto \langle t_D^{\mathcal{Q}}, u\rangle$ over the domain specified by those 13, respectively 154, inequalities. Note that the formula for the data vector relative to BIC is also known, see (8.39) in (Studený, 2005):

$$t_D^{\mathsf{BIC}}(A) = d\cdot H(\hat{P}_A|\prod_{i\in A}\hat{P}_i) - \frac{\ln d}{2}\cdot\{\,|A| - 1 + \prod_{i\in A} r(i) - \sum_{i\in A} r(i)\}$$

for $A \subseteq N$, where $H(*|*)$ is the relative entropy and $\hat{P}_A$ is the marginal empirical distribution given by (the projection of the database) D_A.

5 Geometric Neighborhood

We say that two standard imsets $u, v \in S$ are *geometric neighbors* if the line-segment E connecting them in $\mathbb{R}^{\mathcal{P}(N)}$ is an edge of the polytope P (generated by S), which means $\mathsf{P} \setminus E$ is convex. The motivation for this concept has already been explained in the Introduction. Of course, the concept of geometric neighborhood can be extended to the corresponding BN structures, and to the essential graphs as well.

[4] To avoid misunderstanding recall that distinct $G, H \in \mathsf{DAGS}(N)$ may give the same standard imset $u_G = u_H$; however, the set S contains only one imset for each independence equivalence class.

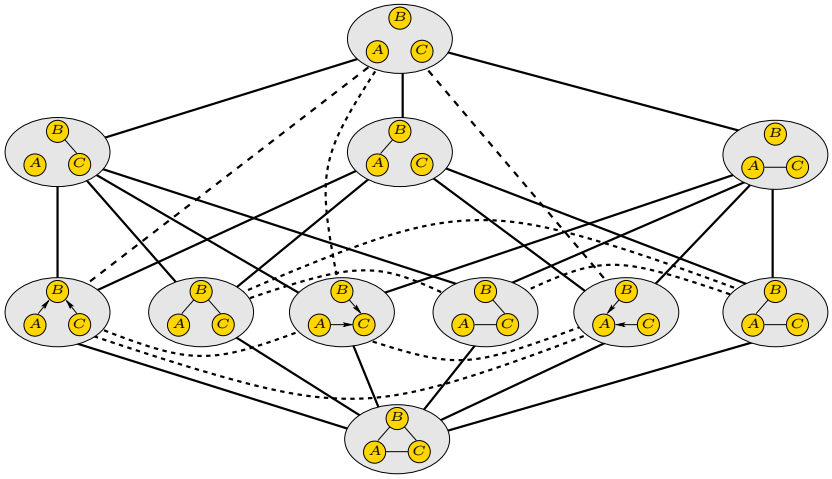


Figure 1: The geometric and inclusion neighborhood (for essential graphs) in the case of 3 variables.

EXAMPLE We characterized the geometric neighborhood in the case of three variables and compared it with the inclusion neighborhood. We found out that the inclusion neighborhood is contained in the geometric one. The result is depicted in Figure 1, in which BN structures are represented by essential graphs, solid lines join inclusion neighbors and dashed lines geometric neighbors that are not inclusion neighbors. Different levels correspond to the numbers of edges.

We made a similar computation also in the case of four variables – see vertex-vertex table in (Vomlel and Studený, 2008). The description of our method for computing the geometric neighborhood is also available at (Vomlel and Studený, 2008).

5.1 GES Failure

What does it mean that $u, v \in S$ are geometric but not inclusion neighbors? The fact that they are geometric neighbors means there exists a linear function on $\mathbb{R}^{\mathcal{P}(N)}$ achieving its maximum over S just in $\{u, v\}$. Analogously, since u is a vertex of P, there exists (another) linear function achieving its maximum just in u. Therefore, by a suitable convex combination of these functions, one can construct a linear function L on $\mathbb{R}^{\mathcal{P}(N)}$ such that $L(u) > L(v) > L(w)$ for any $w \in S \setminus \{u, v\}$. Provided u and v are not inclusion neighbors, L achieves its local maximum (with respect the inclusion neighborhood) in v and the global maximum over S in u.

Now, it has already been explained that every "reasonable" quality criterion $\mathcal{Q}$ is (the restriction of) an affine function on $\mathbb{R}^{\mathcal{P}(N)}$. Thus, the reader may ask whether this may happen for $\mathcal{Q}$ in place of L. Indeed, this is true in the case of three variables for the imset $u = u_{\langle a,c|\emptyset\rangle}$, which corresponds to an "immorality" $a \to b \leftarrow c$ and the imset v corresponding to the empty graph – see Figure 1.

EXAMPLE There exists a database D (of the length $d = 4$) over $N = \{a, b, c\}$ such that the BIC criterion achieves its local maximum in the empty graph G^0 and its global maximum in (any of) the graph(s) $\widehat{G}$ of the type $a \to b \leftarrow c$. Put $\mathsf{X}_i = \{0, 1\}$ for $i \in N$ and $x^1 = (0, 0, 0)$, $x^2 = (0, 1, 1)$, $x^3 = (1, 0, 1)$, $x^4 = (1, 1, 0)$. Then direct computation of BIC (see § 2.1.2) gives $\mathsf{BIC}(\widehat{G}) = -14 \ln 2$, $\mathsf{BIC}(G^0) = -15 \ln 2$ and $\mathsf{BIC}(G') = -16 \ln 2$ for any graph G' over N having just one edge.

The reader may object that this is perhaps a rare casual example because of a short database. However, BIC exhibits the same behavior if the database D is multiplied! The limited scope of this contribution does not allow us to give the arguments why (we think) this is, actually, asymptotic behavior of any consistent score equivalent decomposable criterion $\mathcal{Q}$, provided the database is "generated" from the empirical

distribution $\hat{P}$ given by D. The point is that $\hat{P}$ is **not** perfectly Markovian with respect to any $G \in \mathsf{DAGS}(N)$.

In particular, the GES algorithm – see (Chickering, 2002) for details about this algorithm – should (asymptotically) learn the empty graph G^0, while it is clear that (any of the graphs) $\widehat{G}$ is a more appropriate BN structure approximation of the "actual" conditional independence structure given by $\hat{P}$.

6 Conclusion

In our view, this is an example of the failure of the GES algorithm which may occur whenever a disputable *data faithfulness assumption* is not fulfilled.[5] This assumption is "valid" if data are artificially generated, but, in our view, one can hardly ensure its validity for "real" data.

On the other hand, the point of the example from 5.1 is that the GES algorithm is based on the inclusion neighborhood. This cannot happen if the greedy search technique is based on the geometric neighborhood. Indeed, we are able to show that each local maximum (of an affine function) with respect to the geometric neighborhood is necessarily a global maximum (over P). The proof is at the manuscript stage and will be published later. The conjecture that the inclusion neighborhood is always contained in the geometric one has recently been confirmed by Raymond Hemmecke (personal communication). Therefore, we think the concept of geometric neighborhood is quite important. We plan to direct our future research effort to algorithms for its efficient computation.

Acknowledgements

We are grateful to our colleague Tomáš Kroupa for his help with computations. This research has been supported by the grants GAČR n. 201/08/0539 and MŠMT n. 1M0572, and n. 2C06019.

[5] By this we mean the assumption that data are "generated" from a distribution which is **perfectly Markovian** with respect to an acyclic directed graph.

References

S.A. Andersson, D. Madigan and M.D. Perlman. 1997. A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25:505-541.

R.R. Bouckaert. 1995. Bayesian belief networks: from construction to evidence. PhD thesis, University of Utrecht.

D.M. Chickering. 2002. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507-554.

M. Franz. 2006. Convex – a Maple package for convex geometry, version 1.1, available at `http://www-fourier.ujf-grenoble.fr/~franz/convex/`

M. Frydenberg. 1990. The chain graph Markov property. *Scandinavian Journal of Statistics*, 17:333-353.

S.L. Lauritzen. 1996. *Graphical Models*. Clarendon Press.

C. Meek. 1997. Graphical models, selecting causal and statistical models. PhD thesis, Carnegie Melon University.

R.E. Neapolitan. 2004. *Learning Bayesian Networks*. Pearson Prentice Hall.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

A. Schrijver. 1986. *Theory of Linear and Integer Programming*. John Wiley.

G. Schwarz. 1978. Estimation the dimension of a model. *The Annals of Statistics*, 6:461-464.

M. Studený. 2005. *Probabilistic Conditional Independence Structures*. Springer-Verlag.

M. Studený and J. Vomlel. 2008. Geometric view on learning Bayesian network structures. A draft available at `http://staff.utia.cas.cz/studeny/c11.html`

T. Verma and J. Pearl. 1991. Equivalence and synthesis of causal models. In *6th Conferece on Uncertainty in Artificial Intelligence*, pages 220–227.

J. Vomlel and M. Studený. 2008. Geometric neighborhood for Bayesian network structures over three and four variables. Web page, see `http://www.utia.cas.cz/vomlel/imset/polytopes-3v-and-4v.html`

An Influence Diagram framework for acting under influence by agents with unknown goals

Nicolaj Søndberg-Jeppesen and Finn Verner Jensen
Department of Computer Science
Aalborg University
9220 Aalborg, Denmark

Abstract

We consider the situation where two agents try to solve each their own task in a common environment. We present a general framework for representing that kind of scenario based on Influence Diagrams (IDs). The framework is used to model the analysis depth and time horizon of the opponent agent and to determine an optimal policy under various assumptions on analysis depth of the opponent. Not surprisingly, the framework turns out to have severe complexity problems even in simple scenarios due to the size of the relevant past. We propose an algorithm based on Limited Memory Influence Diagrams (LIMIDs) in which we convert the ID into a Bayesian network and perform single policy update. Empirical results are presented using a simple board game.

1 Introduction

It is a central problem in Multi-agent research to model the reasoning necessary when multiple agents, each with individual objectives, interact in the same environment. While each agent may change the state of the environment towards a more favorable state for itself, other agent's actions may change the state to a less favorable state. When planning under such conditions it is beneficial to take into account the other agent's reasoning. The Recursive Modeling Method (RMM) which was proposed by Gmytrasiewicz et al. (1991) does that. They propose to equip each intelligent agent with a model in which each agent is equipped with a model which models the other agents. These nested models may again have models of all the rest of the agents in the environment which again contain nested models. The nesting of models continues until a predefined nesting level is met. At the deepest level the nesting is ended by a simpler kind of model which equip each agent in the environment with a "flat" model, without models of other agents.

We shall use RMM together with Influence Diagrams (IDs) for modeling a game scenario. In this scenario each agent intends to solve a number of tasks or assignments. The characteristics of the scenario is, that since the agents co-exist in the same environment, the actions performed by one agent affects the state of the scenario for all agents. The scenario may be a competition between the agents, they may cooperate in solving the same task or they may be working on solving each their task without caring about the other agent's performance. No matter what, the success for each agent is highly dependent on its ability to model the other agents in the scenario.

Many of the RMM based approaches turn out to be PSPACE-hard (Gmytrasiewicz and Doshi, 2005). IDs are also facing notorious complexity problems due to the no-forgetting assumption (the assumption that everything that was known at a previous decision is also known at the current decision). Eventually the decision maker will have way too much information all of which being relevant for the current decision. Lauritzen and Nilsson (2001) propose Limited Memory Influence Diagrams (LIMIDs) in which the decision maker is assumed to have only a certain amount of memory. They propose an al-

gorithm called *single policy update*, which finds an approximation to the optimal policy. In this paper we shall propose an algorithm which is able to solve RMM based LIMIDs in multiagent scenarios.

2 Background

The kind of scenarios we are interested in consist of a finite set of *world states* $\boldsymbol{W}$ with states $w_1, w_2, \ldots, w_m$, and 2 *agents* P^1 and P^2. We assume P^1 to be female and P^2 to be male. The agents have finite sets of actions, say $\boldsymbol{Actions_{P^1}}$ and $\boldsymbol{Actions_{P^2}}$ with members $action^1_{P^1}, action^2_{P^1}, \ldots, action^k_{P^1}$ and $action^1_{P^2}, action^2_{P^2}, \ldots, action^k_{P^2}$ respectively. The transition between world states at time t to time $t+1$ is determined by a probabilistic function τ, where $\tau : \boldsymbol{W} \times \boldsymbol{Actions_{P^1}} \times \boldsymbol{Actions_{P^2}} \times \boldsymbol{W} \rightarrow \boldsymbol{[0;1]}$.
Furthermore, each agent has an assignment which reflects how much the agent prefers each world state by assigning a value to each state. Thus, agent P^1 and P^2, have a finite set of possible assignments $a_{P^1_1}, a_{P^1_2}, \ldots, a_{P^1_l}$ and $a_{P^2_1}, a_{P^2_2}, \ldots, a_{P^2_m}$ respectively. We will consider only scenarios where the world state is always known by all agents but the actual assignments of the other agents remain hidden. A probability distribution of the opponent's assignment can however, be obtained by observing his/her actions.
You may consider the scenario as a board game, where each player wishes to obtain certain pattern on the board. The players receive their pattern assignments by drawing cards from a deck. The payoff function, which assigns payoffs to each player at each time step, is a function of the current world state and the assignment of the two agents. We shall refer to the scenario as *covert interference*(CIF).

2.1 Influence Diagrams

We shall use the classical paradigms from Probabilistic Graphical Models (PGMs). A graphical model is a directed acyclic graph with three types of nodes, *chance nodes* (circular nodes), *decision nodes* (rectangular nodes), and *utility nodes* (diamond shaped nodes). A directed link into chance node reflects (causal) impact, which may be of non-deterministic character, a link into a decision node represents information. That is, if C is a parent of the decision node D then the state of C is known by the decision maker when D is to be decided.

The quantitative part of a PGM consists of utility functions and conditional probabilities. For a utility node U with parents $pa(U)$ we specify the utility as a function of $pa(U)$. For a chance node C with parents $pa(C)$ we specify $P(C|pa(C))$, the conditional probability of C given $pa(C)$.

A *solution* to an ID is an *optimal strategy*. A strategy consists of set of *policies*, one for each decision node. A policy for a decision node is a function, which given the known past provides a decision. A strategy is *optimal* if it maximizes the decision maker's expected utility.

There are standard algorithms for solving IDs, and systems for specifying and solving IDs are commercially available (Shachter, 1986; Shenoy, 1992; Jensen et al., 1994; Hugin Expert A/S, 2007). We shall in this paper take these algorithms for granted.

The framework of IDs has been extended in various ways. In particular, Koller and Milch (2003) introduced *Multi-Agent Influence Diagrams* (MAIDs). The various acting agents are given decision and utility nodes of particular colors (or shadings).

2.2 Graphical representation of CIF

We adapt the framework of MAIDs to CIF. This is illustrated in Figure 1. In Figure 1 player P^1's nodes are lightly shaded, and P^2's nodes are darkly shaded.

The nodes $W_0, W_1, \ldots$ represent the world states at $t = 0, t = 1, \ldots$ the chance nodes A^1 and A^2 represent the players' assignments, the nodes with P^1-labels represent the moves by player P^1. The diamond shaped nodes, which are half lightly shaded and half darkly shaded represent the payoff matrixes, which assign a utility to both players in each game step. The links from a W-node and the A-nodes to a U-node indicate that the utility is a function of the world state and the assignments. The links

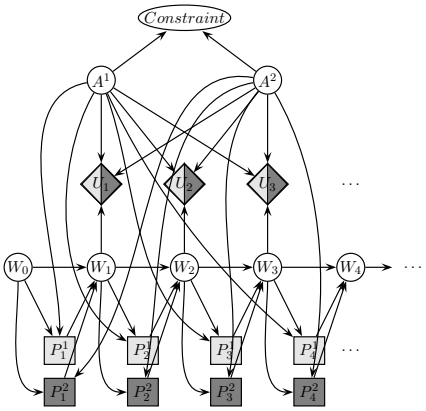


Figure 1: A MAID representation of CIF.

from the A^1-node to decision nodes represent that player P^1 knows her assignment. The links from W-nodes to decision nodes represent that the state of the world is always known. The dots at the right of the graph indicate that there is no time limit specified.
There might be constraints on which assignments the players can have simultaneously e.g. they might not be able to have the same assignment. Therefore, A_1 and A_2 are connected to a constraint node which is instantiated.

At $t = 0$ the game starts in W_0, where P^1 and P^2 each decide their actions concurrently, knowing only their own assignment and the initial world state. P^1's and P^2's joint moves lead to a new state W_1, where player P^1 and P^2 again decide each their actions knowing only W_1 and each their own assignments. When both players have decided their actions the game continues with time $t = 1$. At $t = 1$ P^1 still knows only her own assignment but if she knows P^2's policy she can estimate P^2's assignment in A^2.

Even if there is a pre-specified time horizon, the standard methods for solving IDs ((Shachter, 1986; Shenoy, 1992; Jensen et al., 1994)) cannot be used. Consider the last time step. Both players have to come up with an optimal decision given the past. Part of the considerations for player P^1 will be an estimate of player P^2's move. However, P^2's move is dependent on an estimate of P^1's move. You end up with an infinite regression, which in game theory is solved by determining Nash equilibria (Nash, 1950).

We consider the situation, where we wish to construct a computer program to play against human players. As we cannot expect human players to play Nash equilibria, the computer shall exploit that, and therefore it usually shall not play Nash-equilibria either.

2.3 The game seen in the eyes of P^1

In real world situations, players do not perform an infinite regression and determine Nash equilibria. The players will analyse the situation to a certain depth and with a certain lookahead of moves, and in the depth analysis they will make some assumptions about the other players' analysis depth and look-ahead. This is called the *recursive modeling method* (RMM) (Gmytrasiewicz et al., 1991), and we will incorporate RMM into the models by letting P^1 bound the recursive modeling to a certain level. More specifically, P^1 has a model incorporating the moves of player P^2, where she makes some assumptions on how many moves ahead he analyzes the situation, and in turn, how deep P^2 is assuming P^1's model to be. In other words, if P^1 makes these assumptions about the policies of P^2, the model in Figure 1 can be transformed to an ID, where P^2's decision nodes are replaced with chance nodes, and $P(P^2_{i+1}|A^2, W_0, \ldots, W_i, P^2_0, \ldots, P^2_i)$ is the policy (see Figure 2). Note that the node A^2 reflects that P^2's assignment is unknown to P^1. The model shall include prior probabilities for A^2.

In the simplest case P^1 assumes that P^2 just picks a move randomly. We will say that this player has a level 0 model since she in this case uses the least effort to model P^2. In case P^1 assumes that P^2 has a level 0 model, we say that P^1 has a level 1 model. In general, when P^1 has a level i model, she assumes that P^2 has a level $i-1$ model. As we let P^1 be the computer and P^2 the human, we assume P^1 to have a larger analysis depth than P^2.

At each level, P^1 may take different numbers of future time steps into account. If P^1 is only taking one future time step into account she will greedily pick a move that maximizes her

expected utility in the next time step. If P^1 is taking 2 future time steps into account she will maximize the sum of her expected utility in the next and the following time step. In general, if P^1 is taking h future time steps into account she will maximize her expected sum of utility in the next h time steps. We shall call the number of future time steps P^1 takes into account P^1's *time horizon*. Consequently P^1 also must have an assumption about P^2's time horizon and she must also have an assumption about which time horizon P^2 assumes that P^1 has etc.

In order to capture P^1's modeling level and time horizon together with her assumptions about P^2's nesting depth and P^2's assumptions about P^1's nesting depth we give the following definition.

Definition 1. A player P is a pair defined as follows:

1. $P = (h, NIL)$ is a player with time horizon h and modeling level 0.
2. Given a player O, with modeling level $i-1$, $P = (h, O)$ is a player with time horizon h and modeling level i.

Thus, the simplest model, which is a level 0 model with time horizon 1 is denoted $(1, NIL)$; a $(2, (1, NIL))$ model is a level 1 model in which P^1 has time horizon 2 assuming that P^2 is a level 0 model with time horizon 1; a $(3, (2, (1, NIL)))$ model is a level 2 model in which P^1 has time horizon 3 assuming that P^2 is a level 1 model with time horizon 2 assuming that P^1 is a level 0 model with time horizon 1. Note that, it is possible to define models in which P^1 assumes that P^2 has a longer time horizon than herself. Scenarios where players want to maximize a short time gain playing against players with a long time horizon are common at, for example, stock exchanges.

Figure 2 shows how a $(2, (2, (1, NIL)))$ model for P^1 is represented as an ID. The leftmost ID represents the world as seen by P^1. In this ID, the nested model, namely $(2, (1, NIL))$ is used to fill in the conditional probability distributions in the nodes P_1^2 and P_2^2 representing P^2's decisions. The ID in the center represents this model, which represents the strategy assumed to be played by P^2. In this model the conditional probability distributions P_1^1 and P_2^1 are found by analyzing the rightmost ID, which represents the deepest model $(1, NIL)$. In the $(1, NIL)$ model the chance node P_1^2 represents the completely random strategy which P^2 is assumed to play on level 0. Note that the leftmost ID in Figure 2, contains extra arcs, namely the arc connecting the nodes P_1^2 and P_2^2 and the arc connecting W_0 and P_2^2. These arcs represent that P^2 in time step 1 when P^2 is taking decision P_2^2 will remember W_0 and P_1^2 due to the no-forgetting-assumption (the assumption that whatever was known when taking a decision at time $i < t$ is also known at time t).

The not-forgotten information can be used to *learn* the opponent's assignment. Look at the ID in the center in Figure 2. At t=1 P^2 can use his model of P^1 to learn P^1's assignment in A^1. P^2 will calculate, $P(A^1|W_0, W_1, P_1^2)$ which can be found using the the ID. Thus, the no forgetting assumption has an impact as P^2 can be assumed to have learned something about the state of A^1.

The modeling of the learning opponent causes the dimensionality of the conditional probability table representing P_i^2 to grow heavily with the number of future world states to take into account. In this work we are addressing this problem by reducing the complexity introduced by the no-forgetting assumption in IDs.

3 Agents with limited memory

The no-forgetting assumption results in notorious complexity problems for IDs. In the study reported in (Søndberg–Jeppesen and Jensen, 2008) we used IDs directly in situations like the model in Figure 2 and we ran into serious complexity problems. In our framework we have seen that not only does it impact the decision maker when solving the ID, it also causes the chance nodes representing the opponent's decisions to have intractably large domains.
To address this problem we apply LIMIDs which provide a way to address the complexity problems in IDs by assuming that the decision maker

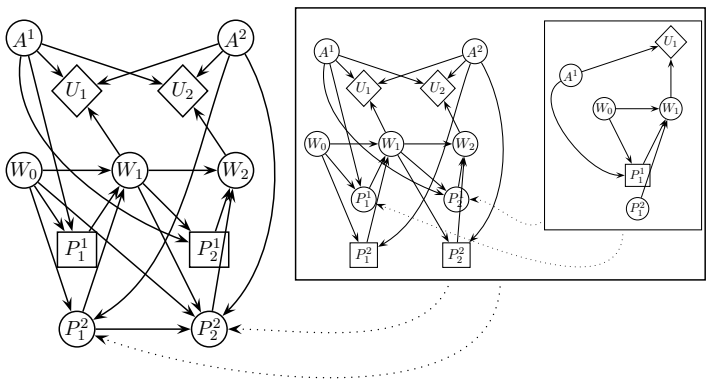


Figure 2: An ID representing the $(2,(2,(1,NIL)))$ model.

has limited memory (Lauritzen and Nilsson, 2000). Syntactically, LIMIDs are like IDs with the only difference that the only things known at a decision node D are nodes with arcs going into D, $(pa(D))$. This means that all the information that the decision maker is assumed to remember when making the decision must be expressed explicitly by information arcs. In our framework we introduce a LIMID player as a player with a certain memory. Like a regular player from Definition 1 a LIMID Player has a time horizon and a model of the opponent, however the LIMID Player also has a certain memory. If P^1 is a LIMID player with memory m, she will at decision P_i^1 remember her previous m decisions $P_{i-m}^1 \dots P_{i-1}^1$, together with the previous m world states $W_{i-m} \dots W_{i-1}$.

Definition 2. A LIMID player L is a triple defined as follows:

1. $L = (h, m, NIL)$ is a LIMID player with time horizon h, memory m and modeling level 0.

2. Given a player or a LIMID player O, with modeling level $i-1$, $L = (h, m, O)$ is a LIMID player with time horizon h, memory m, and modeling level i.

The second entry allows LIMID players to assume that their opponents are ordinary players.

Single policy updating is an iterative algorithm which is guaranteed to converge towards a local maximum policy (Lauritzen and Nilsson, 2000).

Before we describe how the single policy updating algorithm works, we will describe how a policy can be found at each decision by first converting an ID into a Bayesian network.

3.1 Finding optimal policies

We convert an ID into a Bayesian network by converting the utility nodes and the decision nodes into chance nodes. This method was first proposed by (Cooper, 1988).

First replace each decision node P_i^1 with a chance node with the parents $pa(P_i^1)$, the relevant information nodes for the decision. The possible outcome of this new node is the possible decisions in the decision node. Eventually, this node shall receive a probability distribution representing an optimal policy at this node. Initially the node has uniform priors $P(P_i^1 = d|pa(P_i^1)) = 1/N_i$ for all decisions d in P_i^1 where N_i is the number of possible decisions.

Next, each Utility node U_i is replaced by a two-state chance node NU_i with the same set of parents but with possible outcomes y and n. For convenience we will write nu_i instead of $NU_i = y$ and $\overline{nu}_i$ instead of $NU_i = n$ throughout the rest of the paper. We need to scale the utility values from U_i into the interval $[0;1]$. Let $\mathcal{X}_{pa(NU_i)}$ denote the possible configurations of the variables in the set $pa(NU_i)$, with $x_{pa(NU_i)} \in \mathcal{X}_{pa(NU_i)}$. Now nu_i receives its probability distribution according to the function:

$$P(nu_i|x_{pa(NU_i)}) = \frac{U_i(x_{pa(NU_i)}) - u_{min}}{u_{max} - u_{min}}, \quad (1)$$

where $U_i(x)$ is the utility value for a configuration x according to U_i, while u_{max} and u_{min}

are the maximum and minimum values of U_i. Since we have assumed that the utility function is the same for all U_i the sum of normalized utilities and the sum of utilities are maximal for the same decision.

We shall show how this Bayesian network can serve to find an optimal policy.

Theorem 1. *Given a player with time horizon h and ID $\mathcal{I}$ which has been converted to a Bayesian network $\mathcal{B}$, and let $\boldsymbol{e} = pa(P_i^1)$. The optimal policy $\delta_{P_i^1}(pa(P_i^1))$, can be determined in the following way,*

- *If P_i^1 is the last decision (i.e. $i = h$)*

$$\delta_{P_i^1}(\boldsymbol{e}) = \operatorname*{argmax}_d P(d, |nu_i, \boldsymbol{e}). \quad (2)$$

- *If P_i^1 is not the last decision (i.e. $i < h$) but the CPTs in $P_{i+1}^1, \ldots, P_h^1$ have been replaced with their optimal policies, then $\delta_{P_i^1}(pa(P_i^1))$ is:*

$$\delta_{P_i^1}(\boldsymbol{e}) = \operatorname*{argmax}_d \Big(\sum_{j=i}^{h} P(d, \boldsymbol{e}|nu_j) P(nu_j|\boldsymbol{e}) \Big). \quad (3)$$

The proof for Theorem 1 is a trivial rewrite of the original in (Cooper, 1988) and has been omitted in this paper.

After junction tree propagation with evidence nu_i, the clique containing P_i^1 will hold $P(P_i^1, \boldsymbol{e}, nu_i)$. Hence,

$$\delta_{P_i^1}(\boldsymbol{e}) = \operatorname*{argmax}_{P_i^1} \frac{P(P_i^1, \boldsymbol{e}, nu_i)}{P(\boldsymbol{e}, nu_i)}.$$

When you have an ID with only one decision, then Theorem 1 can be used for an efficient calculation of an optimal policy. We will exploit this in the next section.

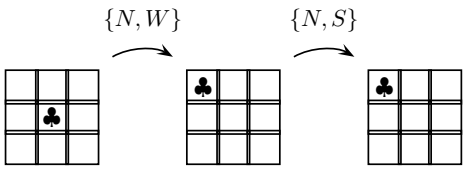
3.2 A LIMID framework

Single policy updating for LIMIDs consists in a series of policy estimates for IDs with one decision. The decision nodes are represented as chance nodes with the information parents as parents. You start off with some policy for all decisions, and then you systematically update the policies for each decision. In each iteration, each local policy $\hat{\delta}_{d_i}(pa(d_i))$ is calculated according to the above trick with the last decisions first. The algorithm iterates until convergence (i.e. no policies change in two consecutive iterations) or for a predefined number of iterations.

- Given a LIMID Player with horizon h, memory m and ID $\mathcal{I}$.
- Convert $\mathcal{I}$ into a Bayesian network $\mathcal{B}$.
- for each decision node P_i^1 in $\mathcal{B}$,
  - add $P_{i-2}^1, P_{i-3}^1, \ldots, P_{i-m}^1$ as parents to P_i^1, and
  - add $W_{i-2}, W_{i-3}, \ldots, W_{i-m}$ as parents to P_i^1.
- Repeat until convergence:
  - For each decision P_i^1, for $i = h, h-1, \ldots, 0$:
  - update $\hat{\delta}_{P_i^1}$ according to Theorem 1.

4 Experimental results

In order to measure the performance of our proposed algorithm we present a simple game which we call Grid. Grid is played by two players P^1 and P^2 that are moving the same single piece on an $m \times m$ grid. Initially, both players are randomly given an assignment, which is a reward function for the positions in the grid. The players may have the same assignment. In each turn the players observe the position of the piece (i.e. the state of the game board) and decide to move it either up, down, right or left, (N, S, E and W). The actions the players have chosen are carried out simultaneously and the resulting effect on the piece is the combination of the two players' moves. If both actions can be carried out (i.e. the resulting position of the piece is still inside the $m \times m$ grid), the piece is first moved to the neighboring cell in the direction of P^1's decision and then to the neighboring cell in the direction of P^2's decision. If at least one of the actions cannot be carried out, the single



Assignment 1			Assignment 2		
8.55	6.82	1.15	-1.66	0.42	6.84
-6.82	0.0	4.12	-0.51	0.0	-0.42
-8.55	-1.15	-4.12	-6.84	0.51	1.66

Figure 3: An example of the game Grid. In the first move, P^1 chooses to move N while P^2 chooses to move W. In the second turn, P^1 and P^2 moves N and S respectively, cancelling each other's effect.

action which can be carried out (if any) is carried out. When the piece is in its new position, the players are rewarded according to their assignment and the next turn begins. The game is a competition so P^1 is punished when P^2 is rewarded and vice versa. This is obtained by subtracting P^2's reward from P^1's reward and vice versa. The game continues for a predetermined number of turns.

An example of Grid with $m = 3$ and 2 possible assignments is shown in Figure 3. Initially, player P^1 and P^2 are assigned Assignment 1 and Assignment 2 respectively. In the first move, P^1 decides to move N while P^2 decides to move W. From the resulting new state, P^1 gets the reward 8.55 while P^2 gets the reward -1.66. Now the scores are $8.55 - -1.66 = 10.21$ to P^1 and $-1.66 - 8.55 = -10.21$ to P^2. In the second move P^1 decides to move N while P^2 decides to move S in which case the board state is not changed. The scores are now 20.42 to P^1 and -20.42 to P^2.

We have implemented Grid together with our proposed algorithm using (Hugin Expert A/S, 2007). The experiments have been performed on a 1.6 GHz PC with 1 GB of RAM.

4.1 Comparison of memory limitations

In the first experiments we have investigated how deep a time horizon players are allowed to get on our system. We start with a 3×3 instance of Grid with 5 assignments and create both a player with a regular ID and a LIMID player to see which values of h we can use in the models before our system runs out of memory. The results are summarized in Table 1. As expected the LIMID allows significantly larger values of h than IDs do.

Table 1: The maximal time horizons possible on our system with different sizes of the Grid game.

Board	ID max h	LIMID max h ($m = 1$)
3×3	4	32
5×5	3	8
7×7	3	8
9×9	2	8

Table 2: Average scores and standard deviations (σ) after 100 Grid games between different models against (2,(2,(1,NIL))).

	Model	P^1	σ
1	(2,(2,(2,(1,NIL))))	5.03	*10.1*
2	(2,1,(2,(2,(1,NIL))))	-0.248	*8.66*
3	(3,(2,(2,(1,NIL))))	7.32	*10,7*
4	(3,2,(2,(2,(1,NIL))))	0.252	*8,27*

4.2 Performance of LIMID players compared to regular players

In a second experiment we have measured how well a LIMID player plays compared a normal player with the same time horizon. We performed experiments with a 3×3 instance of Grid with 5 assignments. Each model has played 100 games of each 10 moves as player P^1 against (2,(2,(1,NIL))) who played as player P^2. The results are summarized in Table 2. When P^1 is playing with the ID models she outperforms P^2 (rows 1 and 3). This is no surprise since she uses more advanced models than P^2. With the LIMID models however (rows 2 and 4) P^1 looses slightly when she plays with lookahead 2 and memory 1 while she wins slightly when she plays with lookahead 3 and memory 2. The σ column indicates a huge variation in the game outcomes.

Table 3: Average scores and standard deviations (*italics*) obtained by players with $h = 3$ on different levels in a 3×3 instance of Grid.

Level	0	1	2	3	4
1	2.47	–	–	–	–
	5.41	–	–	–	–
2	-1.83	3.24	–	–	–
	6.39	*10.76*	–	–	–
3	-3.19	-4.50	9.29	–	–
	7.64	*10.9*	*10.5*	–	–
4	0.55	-4.60	-0.73	8.00	–
	7.06	*10.0*	*5.82*	*10.6*	–
5	0.572	1.18	-6.21	4.40	5.78
	6.36	*7.34*	*10.8*	*10.0*	*8.84*

4.3 Comparison of players of different levels

In a third experiment we have investigated what happens if P^1 has a wrong model of P^2. We do that by letting P^1 assume that P^2 is more intelligent than he really is. Table 3 shows the average scores of Grid games between players of different levels all with $h = 3$ on a 3×3 board. Level 0 refers to the (3,NIL) model, level 1 refers to the (3,(3,NIL)) model etc. The numbers in the cells refer to the score of the row player, i.e. in the first row, the (3,(3,NIL)) has scored on average 2.47 points in games against (3,NIL) who has scored on average -2.47. Again the players play 100 games of each 10 moves. Standard deviations on the game outcomes are shown in italics. As expected, players win when they have the correct assumptions about the opponent, whereas it seems to be less optimal to assume that the opponent is more intelligent than he actually is. This is problematic since a player rarely knows exactly how intelligent the opponent is before a game begins. Rather it should be possible to learn the opponent's level of intelligence during play. We will address this problem in future research.

Acknowledgments

We want to thank the staff in the Machine Intelligence Group at the Department of Computer Science at Aalborg University. In particular, we are grateful to Zeng Yifeng for help and valuable comments during this work. We also thank the anonymous reviewers for their useful feedback on this work

References

G. F. Cooper. 1988. A method for using belief networks as influence diagrams. In *Fourth Wrokshop on Uncertainty in Artificial Intelligence*, pages 55–63.

Piotr J. Gmytrasiewicz and Prashant Doshi. 2005. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24:49–79.

P. J. Gmytrasiewicz, E. H. Durfee, and D. K. Wehe. 1991. A decision theoritic approach to coordinating multiagent interactions. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI 91)*, pages 62–68.

Hugin Expert A/S. 2007. Hugin api reference manual version 6.7. http://download.hugin.com/documents/manuals6.7/api-manual.pdf.

F. Jensen, F. V. Jensen, and S. L. Dittmer. 1994. From influence diagrams to junction trees. In R.L. Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 367–374. Morgan Kaufmann.

D. Koller and B. Milch. 2003. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221.

S. Lauritzen and D. Nilsson. 2000. Evaluating influence diagrams using LIMIDs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 436–445.

J. Nash. 1950. Equilibrium points in N-person games. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 36, pages 48–49.

R. D. Shachter. 1986. Evaluating influence diagrams. *Operations Research*, 34(6):597–609.

P. P. Shenoy. 1992. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40(3):463–484.

N. Søndberg–Jeppesen and F. V. Jensen. 2008. Acting under interference by other agents with unknown goals. In *Proceedings of the 10th Scandinavian Conference on Artificial Intelligence (SCAI).*

Arithmetic circuits of the noisy-or models

Jiří Vomlel
Institute of Information Theory and Automation of the ASCR
Academy of Sciences of the Czech Republic
Pod vodárenskou věží 4,
182 08 Praha 8. Czech Republic.
e-mail: vomlel@utia.cas.cz

Petr Savicky
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2,
182 07 Praha 8. Czech Republic.
e-mail: savicky@cs.cas.cz

Abstract

Arithmetic circuits can be used to represent the process of probabilistic inference in Bayesian networks using methods for which the structure and complexity of the process does not depend on the evidence. For example, for the well-known junction tree methods, there is an arithmetic circuit which represents calculation with the same complexity (Darwiche, 2003). However, arithmetic circuits are more flexible and also allow representation of calculations using different types of computational savings, for example when the conditional probability tables of the Bayesian network have a certain local structure (Darwiche, 2002).

In this paper we use the size of arithmetic circuits to compare the effect of preprocessing Bayesian networks with noisy-or gates using parent divorcing and tensor rank-one decomposition. For this purpose, we use the inference methods implemented in Ace by Chavira and Darwiche for several examples of two-layered networks with noisy-or gates (BN2O type networks) in two situations. First, we use Ace on the original network directly, which means that a kind of parent divorcing is used. Second, before the application of Ace the network is preprocessed using a noisy-max decomposition, originally proposed by Díez and Galán and generalized as tensor rank-one decomposition by Savicky and Vomlel. The size of the resulting circuits depends mainly on the size of the largest clique in the triangulated graph of the network. The treewidth of the optimally triangulated graph of the transformed model is provably never larger than the treewidth of the model preprocessed using parent divorcing. Hence, one may expect that tensor rank-one decomposition produces circuits which are usually not larger than the ones from parent divorcing, even if heuristic triangulation is used. Our experiments with Ace confirm this conclusion on average. However, there are also cases where the transformed network provides a significantly larger circuit. Using a better triangulation computed by Hugin instead of the one computed by Ace we reduced the deterioration factor to at most three. This is much smaller than the best improvement factors, which exceed 100.

1 Introduction

Noisy-or models are probably the most popular examples of canonical Bayesian network models. The canonical models differ from general Bayesian network (BN) models in that they have a certain local structure within the conditional probability tables (CPTs). Canonical models were introduced by Pearl in (Pearl, 1988, Section 4.3.2). In literature they are also called causal independence models or models of independence of causal influence (ICI).

A basic task solved by BNs is the probabilistic inference, typically, the computation of all one-dimensional marginals of the joint probability distribution (represented by a BN) given evidence on a subset of network variables. The conditional independence structure of the models enables efficient probabilistic inference using the standard methods, e.g. the junction tree method (Jensen et al., 1990). It was observed by several authors that one can also benefit from the local structure of the CPTs and further improve the computational efficiency of the probabilistic inference.

In this paper we focus on the noisy-or models. A well-known family of noisy-or models are two-level noisy-or networks, abbreviated as BN2O networks. A BN2O network is a BN having the structure of a bipartite graph with all edges directed from one part (the top level) toward the other (the bottom level) and where all CPTs are noisy-or gates. See Figure 1 for an example of a BN2O network structure. An example of a real BN2O network is the decision theoretic version of the Quick Medical Reference model (QMR-DT) (Shwe et al., 1991). This model consists of approximately 600 nodes corresponding to diseases (top level) and 4000 nodes corresponding to findings (bottom level). Each finding has a subset of diseases as its parents. An algorithm tailored for the BN2O networks is the Quickscore algorithm (Heckerman, 1990), where the computations are optimized with respect to evidence on findings.

A different approach that does not assume evidence to be known in advance and that can benefit from the local structure of CPTs is

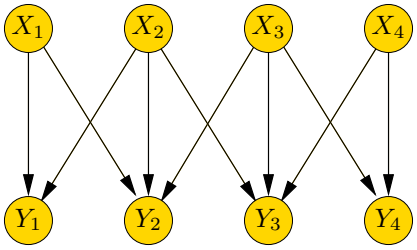


Figure 1: An example of a BN2O model structure.

the compilation of a BN into an arithmetic circuit (AC) (Darwiche, 2003). An AC is a rooted, directed acyclic graph whose leaf (input) nodes correspond to circuit inputs (variables) and whose other nodes are labeled with multiplication and addition operations. The root (output) node corresponds to circuit output. The variables are evidence indicators and parameters of the CPTs. An arithmetic circuit may be used to represent the computation determined by a junction tree (Jensen et al., 1990) in a clustering method, which consists of a fixed sequence of elementary arithmetic operations (additions and multiplications) with real numbers. However, arithmetic circuits are more flexible and may be used to represent different types of computational savings, if they are possible due to specific properties of the initial BN. We use the two methods for constructing an AC implemented in Ace, namely c2d and tabular, see (Chavira and Darwiche, 2006; Chavira and Darwiche, 2007) for more detail. The size of the circuit is used as a measure of complexity of the inference, which is more objective than the running time; the latter is influenced not only by the algorithm, but also by the efficiency of its software implementation.

In this paper we investigate transformations of CPTs representing a noisy-or model before the BN is compiled to a circuit. The standard approach to this problem is parent divorcing (Olesen et al., 1989). We propose to use a transformation based on the decomposition proposed in (Díez and Galán, 2003; Vomlel, 2002). It is a special case of tensor rank-one decomposition of CPTs and, according to the result

of (Savicky and Vomlel, 2007), it uses the minimum possible number of additive components in the case of noisy-max. We show that preprocessing of BN using the above-mentioned transformation before a compiler from BN to AC is used may significantly reduce the size of the resulting circuit obtained by Ace. We systematically tested a range of parameters of artificial networks with randomly generated edges and compared the size of the resulting circuit with parent divorcing as implemented in Ace and with tensor rank-one decomposition. In most cases, the transformed network provided a smaller model. Sometimes, the improvement ratio exceeds 100. There were also cases where the transformed network provided a larger circuit. Analysis of these cases revealed that the tabular method is quite sensitive to the choice of an elimination order of the variables, whereas Ace uses a suboptimal order. Hence, we recalculated some of the cases with the largest deterioration using the optimal order provided by Hugin. Using this more careful method, the transformation never caused an increase of the size of the circuit compared with parent divorcing by a factor larger than 3.

The paper is organized as follows. Necessary notation is introduced in Section 2. Section 3 describes the suggested transformation of a BN as a preprocessing step before an AC compiler is used. In Section 4 we present the necessary information on arithmetic circuits (ACs) and give an example of an AC for the noisy-or gate. In Section 5 we present the experiments which demonstrate the effect of the preprocessing step on the size of the resulting circuit in several randomly chosen examples. Section 6 contains conclusions and a remark on possible directions of future work.

2 Preliminaries

Let N be a set of discrete random variables. The variables will be denoted by capital letters (e.g., X_i, Y, A, etc.) and their states by lowercase letters (e.g., x_{ij_i}, y, a). For both the variables and their states boldface letters will denote vectors.

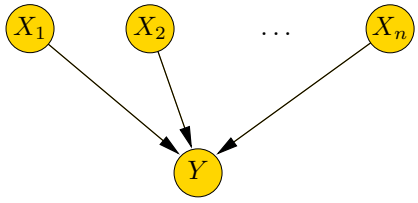


Figure 2: Graph of the noisy-or gate.

Bayesian network (BN) is defined by a pair $\mathcal{N} = (G, \mathcal{P})$, where

- $G = (N, E)$ is an acyclic directed graph (DAG), the nodes of the graph G are variables from the set N, and E is the set of directed edges.
- $\mathcal{P}$ is a system of conditional probability tables $\{P(X|pa(X)),\ X \in N\}$, where $pa(X)$ denotes the vector of parents of X in G.

The joint probability distribution satisfying just the conditional independence statements implied by the DAG and having the probabilities from $\mathcal{P}$ as its (conditional) marginals is uniquely determined by

$$P(\boldsymbol{X}) \ = \ \prod_{X \in N} P(X|pa(X)) \ .$$

A specific example of a BN that we will use in this paper is the noisy-or gate.

Example 1 (Noisy-or gate)**.** Assume a BN of $n+1$ Boolean variables $X_1, \ldots, X_n$ and Y representing the noisy-or relation of Y to $X_1, \ldots, X_n$. The structure of the model is depicted in Figure 2 and the CPT of Y has a local structure defined for $(x_1, \ldots, x_n) \in \{0,1\}^n$ by

$$P(Y = 0|X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} p_i^{x_i} \ , \quad (1)$$

where $p_i \in [0,1], i = 1, \ldots, n$ represent the noise.

3 Tensor rank-one decomposition for noisy-or models

The decomposition suggested in (Díez and Galán, 2003) is applicable to any noisy-max

gate; however, let us describe it only for networks with noisy-or gates. The transformation is applied to each noisy-or gate separately. A noisy-or gate as depicted in Figure 2 is replaced by a subgraph with one additional node B and undirected edges connecting this node with the original nodes as presented in Figure 3. Then, the CPT $P(Y|pa(Y))$ is removed from the network and replaced by $|pa(Y)|+1$ two-dimensional tables for the new edges, which are as follows.

Let $pa(Y) = \{X_1, \ldots, X_n\}$. The interactions between B and X_i and between B and Y are represented by $\varphi_i(B, X_i)$ for $i = 1, \ldots, n$ and $\xi(B, Y)$, respectively, chosen so that for all $(x_1, \ldots, x_n, y) \in \{0,1\}^{n+1}$ we have

$$P(Y = y|X_1 = x_1, \ldots, X_n = x_n) \qquad (2)$$

$$= (1-2y)\prod_{i=1}^{n} p_i^{x_i} + y\prod_{i=1}^{n} 1 \qquad (3)$$

$$= \sum_{b=0}^{1} \xi(b, y) \cdot \prod_{i=1}^{n} \varphi_i(b, x_i) \ , \qquad (4)$$

where the two summands in (3) correspond to the choices $b = 0$ and $b = 1$ in (4). Equality between (2) and (3) follows from (1) and the fact that the sum of (3) for $y = 0$ and $y = 1$ is 1 independently of the values of the remaining variables. Note that the decomposition above uses tables which contain also negative numbers. The original BN may be achieved simply by marginalizing out the auxiliary variable B.

The decomposition is equivalent to the decomposition of $P(Y = y|X_1 = x_1, \ldots, X_n = x_n)$ understood as an $n+1$ dimensional tensor into a sum of tensors of rank one (De Lathauwer and De Moor, 1996). In particular, the minimum number of states of B for which such a decomposition is possible is equal to the rank of $P(Y = y|X_1 = x_1, \ldots, X_n = x_n)$. For noisy-or, this rank is 2, if $p_i < 1$ for at least one index i. Tensor rank-one decompositions are available also for some other canonical models, see (Savicky and Vomlel, 2007).

In Figure 4 we give the transformed structure of BN2O model from Figure 1.

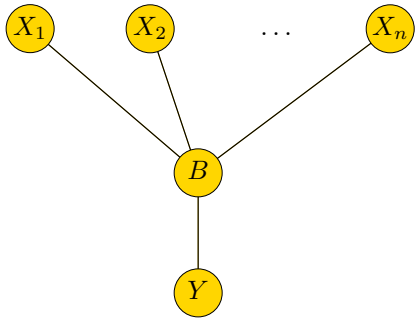


Figure 3: Noisy-or gate from Figure 2 after the transformation using tensor rank-one decomposition

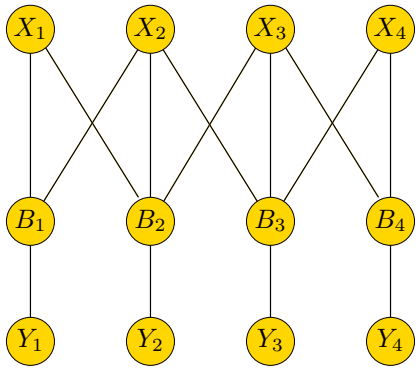


Figure 4: The BN2O model structure from Figure 1 after the transformation using tensor rank-one decomposition.

4 Arithmetic circuits

Let $\boldsymbol{e}$ be evidence $(\boldsymbol{X}_A = \boldsymbol{x}_A^*) = (X_i = x_i^*)_{i \in A}$. Arithmetic circuits as described in the introduction are used to efficiently calculate the probability $P(\boldsymbol{e})$, whose value is given by the multilinear polynomial (Darwiche, 2003)

$$P(\boldsymbol{e}) \quad = \quad \sum_{\boldsymbol{x}} \prod_{X} \lambda_x \ \theta_{x|\boldsymbol{u}} \ ,$$

where the polynomial variables are:

- *BN parameters*: for each variable X and its parents $\boldsymbol{U} = pa(X)$ we have for all values x of X and all values $\boldsymbol{u}$ of $\boldsymbol{U}$ a variable $\theta_{x|\boldsymbol{u}}$, which represents the value

$$\theta_{x|\boldsymbol{u}} \quad = \quad P(X = x|\boldsymbol{U} = \boldsymbol{u})$$

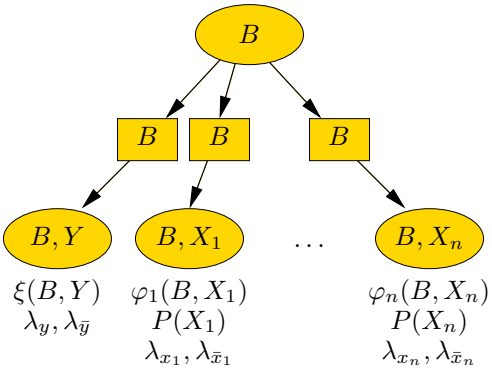


Figure 5: A join tree for the transformed noisy-or gate from Figure 3.

- *evidence indicators*: for each variable X and for each value x of X we have the variable λ_x. The values of the indicator variables encode the given evidence $\boldsymbol{e}$ as follows. The variable λ_x equals one if state x is consistent[1] with the evidence $\boldsymbol{e}$ and zero otherwise. In particular, if there is no evidence for variable X, then $\lambda_x = 1$ for all states x of X.

The structure of the circuit is usually very different from the expression above in order to achieve efficiency.

Example 2 (AC for the noisy-or gate)**.** Let the states of Boolean variables X_i $(i = 1, \ldots, n)$ be denoted x_i (if X_i is true) and $\bar{x}_i$ (if X_i is false) and similarly for the Boolean variable Y.

An AC of a noisy-or gate can be constructed directly from a join tree of the transformed noisy-or gate from Figure 3 (which is a decomposable model) using the construction described in (Darwiche, 2003, Definition 5). We present a join tree[2] of the transformed noisy-or model in Figure 5. Note that each model parameter and each evidence indicator is attached to a node of the join tree.

First, we create one output addition node for the root cluster $\{B\}$ of the join tree. It has as its children two multiplication nodes – one for each state of the variable B. Both multiplication nodes have as their children one addition node from every child separator $\{B\}$ of the root cluster with the compatible state of B. Every addition node from every separator $\{B\}$ has as its children two multiplication nodes from its child cluster ($\{B, Y\}$ or $\{B, X_i\}$ for $i \in \{1, \ldots, n\}$) with the compatible state of B, one for each state of the other variable of the cluster (Y or X_i for $i \in \{1, \ldots, n\}$). Finally, each multiplication node of a leaf cluster has as its children all model parameters and evidence indicators with the compatible states attached to that cluster.

The AC shown in Figures 6 and 7 is the result of the following construction:

1. construct the join tree in Figure 5 for the transformed noisy-or model from Figure 3,
2. use the construction described above to get an AC from the join tree,
3. substitute network parameters from tables $\xi, \varphi_1, \ldots, \varphi_n$ by their values $+1, -1, 0$ and $p_i, i = 1, \ldots, n$, set λ_b and $\lambda_{\bar{b}}$ to 1,
4. simplify the AC by omitting multiplications by one and zero additions, and
5. coalesce parts of the AC that compute the same value so that they are present only once.

The structure of the obtained AC may also be represented by the following formula.

$$\begin{aligned} P(\boldsymbol{e}) \;\; = \;\; & \lambda_y \prod_{i=1}^{n} (\lambda_{\bar{x}_i} \theta_{\bar{x}_i} + \lambda_{x_i} \theta_{x_i}) \\ & + (\lambda_{\bar{y}} - \lambda_y) \prod_{i=1}^{n} (\lambda_{\bar{x}_i} \theta_{\bar{x}_i} + \lambda_{x_i} \theta_{x_i} p_i) \;\; . \end{aligned}$$

Note that the size of this formula is linear in n, while the number of monomials in the expanded polynomial represented by the formula is 2^{n+1}.

If we want to compute all one-dimensional marginals using an AC we can use methods for computing partial derivatives of functions of several variables, e.g., Sawyer (1984). Similarly

[1] Value x of X is consistent with evidence either if $x = x^*$ or if variable X is not present in evidence $\boldsymbol{e}$.

[2] Note that Darwiche's definition (Darwiche, 2003, Section 5.1) of the join tree is more general than the usual one in that it does not require nodes of the join tree to correspond to a maximal clique of the chordal graph.

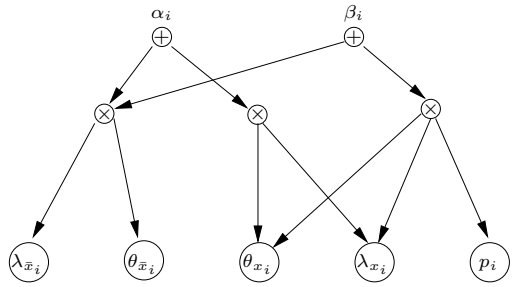


Figure 6: The part of the AC for a noisy-or model corresponding to the variable X_i.

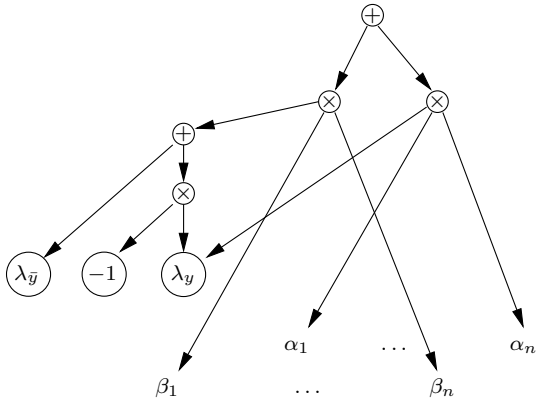


Figure 7: The part of the AC for a noisy-or model that joins the parts of each variable X_i from Figure 6.

to the junction tree methods after two passes through the AC (one upward and one downward pass) we get marginal probabilities for all variables given the evidence $\boldsymbol{e}$. Therefore, in order to get efficient inference it is crucial to have the arithmetic circuit as small as possible. The size of an AC is typically measured by the number of its edges since it is approximately proportional to the number of binary operations.

5 Experimental comparisons

For the experiments we used a development release of Ace (2008). Ace is a package that compiles a BN into an AC using one of two available methods – c2d (Chavira and Darwiche, 2006) or the tabular compilation (Chavira and Darwiche, 2007) – see the Ace manual for details. We carried out experiments with BN2O models of various sizes. The name of the BN2O model in the form of `bn2o-x-y-e-i` contains information about the BN2O structure:

- `x` is the number of nodes in the top level,
- `y` is the number of nodes in the bottom level,
- `e` is the total number of edges in the BN2O model, and `e/y` defines the number of parents for each node from the bottom level.

For each `x-y-e` type ($\texttt{x}, \texttt{y} = 10, 20, 30, 40, 50$ and $\texttt{e/y} = 2, 5, 10, 20$, excluding those with $\texttt{e/y} > \texttt{x}$) we generated randomly ten models (indexed by $\texttt{i} = 0, \ldots, 9$). For every node from the bottom level we randomly selected `e/y` nodes from the top level as its parents. All models and results are available at: `http://www.utia.cz/vomlel/ac/`

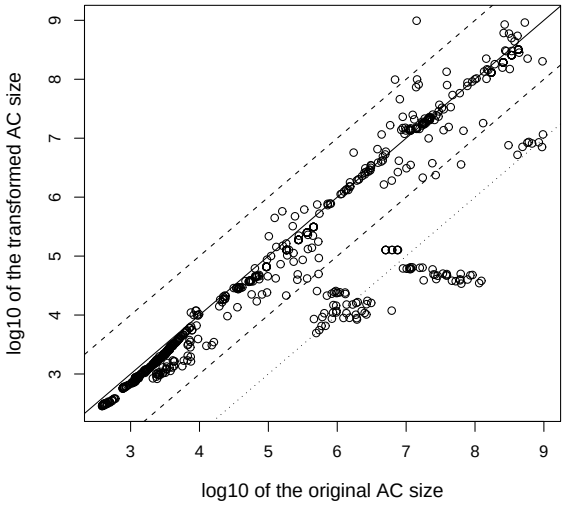


Figure 8: Logarithm of the AC size for the original (o) and transformed (t) models. The straight lines correspond to t/o ratios equal to $10, 1, 1/10, 1/100$.

In Figure 8 we plot the pairs of values of $\log_{10}$ of the ACs' size of the original model[3] (horizontal axis) and the transformed model[4] (vertical

[3]The original model is the model constructed by Ace using parent divorcing.

[4]The transformed model is model obtained using tensor rank-one decomposition.

axis). Since randomness is used during the Ace compilation process, Ace often provides different circuits for the same input model. Therefore every value presented in the plot is the minimum over ten runs – five runs of c2d plus five runs of the tabular method – for both the original and the transformed models.

The AC of the transformed model was smaller in 88% of the BN2O models solved by Ace for both - the original and the transformed models. In several cases we got significant reductions in the AC size (in a few cases multiple order of magnitude), in most of the remaining cases we got smaller reductions. There are also a few cases where the AC of the transformed model is significantly larger. We comment on these cases below. For some of larger models Ace ran out of memory[5] – for 26% of the original models and 21% of the transformed models. For 85% of the tested BN2O models the tabular method led to smaller ACs than c2d.

It is well-known that the efficiency of the inference in BN depends on the size of the largest clique in the triangulated graph of the network or, more exactly, on the total size of the tables in the resulting network. We observed this also in our experiments with Ace; the size of the AC is significantly influenced by the total table size corresponding to the triangulated graph of the models.

The triangulated graph of the transformed model may be obtained from the triangulated graph of the original model by contracting edges between the nodes in the groups of nodes, which are added by parent divorcing for each node in the bottom level. It can be shown that contracting edges does not increase the treewidth of the graph. Hence, the treewidth of the optimally triangulated graph of the transformed model is never larger than the treewidth of the original model. However, if a (non-optimal) heuristic method is used for triangulation then it may happen that we get larger treewidth for the triangulated graph of the transformed model. We believe this is the main reason behind the few large losses of the transformed model.

We conducted additional experiments with all models where Ace provided ACs of the transformed model at least three times larger. In all of these eleven cases we were able to reduce the deterioration factor to less than three using a better triangulation method. Let us have a closer look at a BN2O model having the largest loss for the transformed model – `bn2o-20-30-300-6`. The AC size for the original model was 21 539 918 edges, while the AC of the transformed model generated by Ace when using the minfill heuristics for triangulation had 999 809 342 edges, i.e. about 46 times larger. But, when we constructed the AC of the transformed model using an elimination ordering minimizing the total table size found by Hugin (2008), the AC size dropped to 25 117 892 edges.

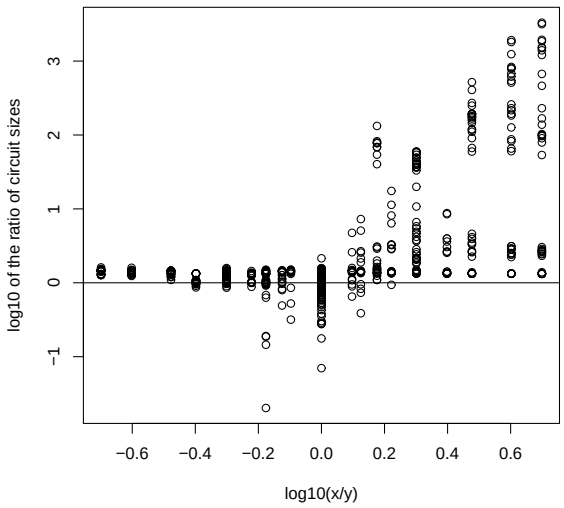


Figure 9: $\log_{10}(o/t)$ with respect to $\log_{10}(x/y)$.

From the plot in Figure 9 we can see that there is a higher probability of larger gains when using the transformed model if there are more nodes in the first level than in the second level of the BN2O network since the log-ratio of the AC sizes of the original and transformed model $\log_{10}(o/t)$ has more often higher values with a positive log-ratio of the number of nodes in the first and the second level $\log_{10}(x/y)$.

[5] All experiments were done with the maximum possible memory for 32 bit Ace, which is 3.6 GB RAM.

We should mention that in (Chavira et al., 2005) the authors perform experiments with BN2O networks, but unlike them we do not assume evidence to be known before the AC is constructed.

6 Conclusions and future work

The performed experiments suggest that tensor rank-one decomposition can help to find smaller ACs for BN2O type networks. In some cases it may even find a solution that could not be found without the transformation. Future work should determine whether similar savings are possible for other canonical models for which a compact tensor rank-one decomposition is known.

Acknowledgments

We are grateful to Mark Chavira for his flexible support of Ace and to Frank Jensen and Anders Madsen for providing us with the optimal triangulation method used in Hugin.

The authors were supported by the Ministry of Education of the Czech Republic under the projects 1M0545 (P. Savicky), 1M0572, and 2C06019 (J. Vomlel). J. Vomlel was also supported by the Eurocores LogICCCC Project FP005 (LcpR) and by the project 201/08/0539 of the Grant Agency of the Czech Republic.

References

Ace. 2008. A Bayesian network compiler. `http://reasoning.cs.ucla.edu/ace/`.

M. Chavira and A. Darwiche. 2006. Encoding CNFs to empower component analysis. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, Lecture Notes in Computer Science, Volume 4121, pages 61–74. Springer Berlin /Heidelberg.

M. Chavira and A. Darwiche. 2007. Compiling bayesian networks using variable elimination. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2443–2449.

M. Chavira, D. Allen, and A. Darwiche. 2005. Exploiting evidence in probabilistic inference. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI), Edinburgh, Scotland*.

A. Darwiche. 2003. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50:280–305.

L. De Lathauwer and B. De Moor. 1996. From matrix to tensor: multilinear algebra and signal processing. In *4th Int. Conf. on Mathematics in Signal Processing, Part I*, IMA Conf. Series, pages 1–11, Warwick. Keynote paper.

F. J. Díez and S. F. Galán. 2003. An efficient factorization for the noisy MAX. *International Journal of Intelligent Systems*, 18:165–177.

D. Heckerman. 1990. A tractable inference algorithm for diagnosing multiple diseases. In *Proceedings of Fifth Conference on Uncertainty in Artificial Intelligence, Windsor, Ontario*, pages 163–171. Elsevier.

Hugin. 2008. Decision engine, version 6.7. `http://www.hugin.com/`.

F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. 1990. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, (4):269–282.

K. G. Olesen, U. Kjærulff, F. Jensen, F. V. Jensen, B. Falck, S. Andreassen, and S. K. Andersen. 1989. A MUNIN network for the median nerve — a case study on loops. *Applied Artificial Intelligence*, 3:384–403. Special issue: Towards Causal AI Models in Practice.

J. Pearl. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

P. Savicky and J. Vomlel. 2007. Exploiting tensor rank-one decomposition in probabilistic inference. *Kybernetika*, 43(5):747–764.

J. W. Sawyer. 1984. First partial differentiation by computer with an application to categorical data analysis. *The American Statistician*, 38(4):300–308.

M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. 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.

J. Vomlel. 2002. Exploiting functional dependence in Bayesian network inference. In *Proceedings of the 18th Conference on Uncertainty in AI (UAI)*, pages 528–535. Morgan Kaufmann Publishers.

Tightly and Loosely Coupled Decision Paradigms in Multiagent Expedition

Yang Xiang and Franklin Hanshar
University of Guelph
Ontario, Canada

Abstract

Frameworks for multiagent decision making may be divided into those where each agent is assigned a single variable (SVFs) and those where each agent carries an internal model, which can be further divided into loosely coupled frameworks (LCFs) and tightly coupled frameworks (TCFs). In TCFs, agent communication interfaces render their subdomains conditionally independent. In LCFs, either agents do not communicate or their messages are semantically less restricted. SVFs do not address the privacy issue well. LCF agents cannot draw from collective knowledge as well as TCF agents. However, disproportional effort has been dedicated to SVFs and LCFs, which can be attributed partially to unawareness of the computational advantages of TCFs over performance, efficiency and privacy. This work aims to provide empirical evidence of such advantages by comparing *recursive modeling method* (RMM) from LCFs and *collaborative design network* (CDN) from TCFs, both of which are decision-theoretic and the latter of which is a graphical model. We apply both to *multiagent expedition* (MAE), resolve technical issues encountered, and report our experimental evaluation.

1 Introduction

We consider frameworks for online decision making (rather than offline policy making, e.g., (Becker et al., 2004)) in cooperative multiagent systems. They may be divided into SVFs where each agent is assigned a single variable in the domain and those where each agent carries an internal model over a subdomain. Frameworks using internal models can be further divided into LCFs and TCFs. In TCFs, agents communicate through messages over agent interfaces that are semantically rigorously defined to render subdomains conditionally independent. In LCFs, either agents do not communicate but rely on observing other agents' actions to coordinate, or their messages are semantically less restricted.

SVFs do not address the issue of private versus public variables well, as they do not have infrastructure to differentiate variables as such. LCFs are often motivated by tasks where agents cannot communicate. Given the proliferation of distributed and wireless computing, it is hard to find task domains where cooperative agents cannot communicate (except a few of military nature). Due to tightly controlled agent interface, joint belief of team agents is well defined and a TCF agent's belief is consistent with the joint belief. This is generally not true in LCFs even when agents do communicate (see Proposition 4.5 and Theorem 8.10 in (Xiang, 2002) for a formal treatment). In other words, a TCF agent draws from collective knowledge better than a LCF agent in general. However, significant research efforts have been dedicated to SVFs, e.g., (Modi et al., 2005; Petcu and Faltings, 2005), and LCFs, e.g., (Gmytrasiewicz et al., 1998; Gmytrasiewicz and Durfee, 2001; Maes et al., 2001; Shen and Lesser, 2006), in comparison with those to TCFs (Xiang, 2002; Xiang et al., 2005). This can be attributed at least partly to unawareness of the computational advantages of TCFs over performance, efficiency and privacy. Hence, empirical evidence of such advantages will contribute to the due adoption of TCFs. In this work, we select one representative, RMM (Gmytrasiewicz et al., 1998), from LCFs and

one, CDN (Xiang et al., 2005), from TCFs, both of which are decision-theoretic and the latter of which is a graphical model. We apply both to the same multiagent decision problem, MAE (Xiang and Hanshar, 2007), resolve technical issues encountered, especially those related to RMM, and compare them experimentally.

Sec. 2 reviews background on MAE, CDN and RMM. Sec. 3 presents technical issues on applying RMM to MAE. Sec. 4 reports experimental results. We discuss the generality issues of this research in Sec. 5.

2 Background

2.1 Multiagent Expedition

We consider MAE in an area represented as a grid of cells. At any cell, an agent can move to an adjacent cell by actions *north*, *south*, *east*, *west* or remain there (*halt*). The effect of an action is uncertain. The desirability of an object (located at a cell) is indicated by a numerical *reward*. A cell that is neither interesting nor harmful has a reward of a *base value*. The reward at a harmful cell is lower than the base value. The reward at an interesting cell is higher than the base value and can be further increased through agent cooperation.

When a physical object at a given location is to be manipulated (e.g., digging), cooperation is often most effective when a certain number of agents are involved, and the per-agent productivity is reduced with more or less agents. Suppose that the most effective level is 2. The reward that can be collected by a single agent from a given cell may be 0.3, and we denote this as a *unilateral reward*. If two agents cooperate at the cell, each receives 0.4, and we denote this as a *bilateral reward*. If three or more agents meet at the cell, two of them each receives 0.4 reward and others receive the base value. This feature promotes effective cooperations and discourages unproductive ones.

After a cell has been visited by any agent, its reward is decreased to the base value. As a result, wandering within a neighborhood is unproductive. Agents have no prior knowledge how rewards are distributed in the area. Instead, at any cell, an agent can reliably perceive the cell's absolute location (e.g., through GPS or triangulation) and reward distribution within a small radius. An agent can perceive the location and communicate with another agent if the latter is within a small radius.

The objective of the agents is to move around the area, cooperate as needed, and maximize the team reward over a finite horizon. They must do so based on local observations and limited communication.

2.2 Collaborative Design Networks

CDN is motivated by collaborative industrial design in supply chains. An agent responsible for a component encodes design knowledge and preference into a *design network* (DN) $S = (V, G, P)$. The *domain* is a set of discrete variables $V = D \cup T \cup M \cup U$. D is a set of *design parameters*. T is a set of *environmental factors* of the product under design. M is a set of objective *performance measures* and U is a set of subjective *utility functions* of the agent.

The dependence *structure* $G = (V, E)$ is a directed acyclic graph whose nodes are mapped to elements of V and whose set E of arcs encode design constraints, dependency of performance on design and environment, and dependency of utility on performance.

P is a set of potentials, one for each node x, formulated as a probability distribution $P(x|\pi(x))$, where $\pi(x)$ are parent nodes of x. $P(d|\pi(d))$, where $d \in D$, encodes a design constraint. $P(t|\pi(t))$ and $P(m|\pi(m))$, where $t \in T, m \in M$, are typical probability distributions. Each utility variable has a space $\{y, n\}$. $P(u = y|\pi(u))$ is a utility function $u(\pi(u)) \in [0, 1]$. Each node u is assigned a weight $k \in [0, 1]$ where $\sum_U k = 1$. With P thus defined, $\prod_{x \in V \setminus U} P(x|\pi(x))$ is a joint probability distribution (JPD) over $D \cup T \cup M$. With the assumption of additive independence among utility variables, the expected utility of a design $\mathbf{d}$ is $EU(\mathbf{d}) = \sum_i k_i(\sum_{\mathbf{m}} u_i(\mathbf{m})P(\mathbf{m}|\mathbf{d}))$, where $\mathbf{d}$ (bold) is a configuration of D, i indexes utility nodes in U, $\mathbf{m}$ (bold) is a configuration of parents of u_i, and k_i is the weight of u_i.

Each supplier is a designer of the supplied component. Agents, one per supplier, form a

collaborative design system. Each agent embodies a design network called a design *subnet* and agents are organized into a *hypertree*: Each hypernode corresponds to an agent and its subnet. Each hyperlink (called *agent interface*) corresponds to design parameters shared by the two subnets, which renders them conditionally independent. They are *public* variables and remaining variables in each subnet are *private*. The hypertree specifies whom an agent can communicate directly. Each subnet is assigned a weight w_i, representing a compromise of preferences among agents, where $\sum_i w_i = 1$. The collection of subnets $\{S_i = (V_i, G_i, P_i)\}$ forms a CDN.

The product $\prod_{x \in V \setminus \cup_i U_i} P(x|\pi(x))$ is a JPD over $\cup_i (D_i \cup T_i \cup M_i)$, where $P(x|\pi(x))$ is associated with node x in a subnet. The expected utility of a design $\mathbf{d}$ is $EU(\mathbf{d}) = \sum_i w_i \ (\sum_j k_{ij} \ (\sum_{\mathbf{m}} u_{ij}(\mathbf{m}) \ P(\mathbf{m}|\mathbf{d})))$, where $\mathbf{d}$ is a configuration of $\cup_i D_i$, i indexes subnets, j indexes utility nodes $\{u_{ij}\}$ in ith subnet, $\mathbf{m}$ is a configuration of parents of u_{ij}, and k_{ij} is the weight associated with u_{ij}. Hence, given a CDN, a decision-theoretical optimal design is well defined. Optimal design (Xiang et al., 2005) is obtained by communicating messages over agent interfaces along the hypertree. After communication, all agents have local designs that are globally optimal (collectively maximizing $EU(\mathbf{d})$). Computation (incl. communication) is linear on the number of agents (Xiang et al., 2005) and is efficient for a non-trivial class of CDNs (Xiang, 2007).

The general problem of MAE is exponentially complex on the number of agents and the length of horizon. A more efficient solution of MAE for limited horizon can be devised based on CDN (Xiang and Hanshar, 2007) (see Fig. 2 in Sec 4). An agent team is divided into groups. It allows group members to cooperate at the most effective level. At the same time, different groups can stay apart so that the area is explored more effectively and planning computation is made more efficient with less group interaction.

Within group, a hypertree organization is imposed to support tightly-coupled communication and reduce agent interaction. For instance, if the most productive level of cooperation is two, a group size of three and an organization $A - B - C$ for agents A, B and C can be used. In each agent subnet, movement actions form design nodes, agent locations form performance nodes, and rewards form utility nodes.

As mentioned above, planning is made more efficient by ignoring inter-group interaction and some inner-group interaction. The computation is sound only if unconsidered interactions do not exist. Such desirable behavior of agents is promoted through modifying the distribution of each utility node. The reward of a location is initialized to the perceived value. If the location is part of a group configuration where intended agent interactions are negatively affected (e.g., group members are too far apart) or unintended interactions are possible (e.g., members of different groups are too close), the reward value will be reduced. With grouping, planning complexity is unchanged as the team size grows.

2.3 Recursive Modeling Method

RMM (Gmytrasiewicz et al., 1998) uses a payoff matrix to encode an agent's preference over consequences of joint actions of team agents. With a total of n agents, a payoff matrix for an agent A has n dimensions, with one corresponding to each agent. The width of each dimension is equal to the number of alternative actions of the agent. Each cell of the matrix corresponds to the outcome of a joint action by all agents and is filled with the sum of rewards.

Agents do not communicate and reason about each other through a hierarchy of models (see Fig. 1 in Sec. 3.2). For instance, in a system with agents A and B, the top level model in A is its own payoff matrix. Each model in the second level represents what A believes to be the payoff matrix of B, assuming a specific state of B. The state is associated with a prior probability of A, and is updated through Bayesian learning (Gmytrasiewicz et al., 1998) based on observed actions of B.

3 Recursive Modeling for MAE

3.1 Payoff Matrix

Grouping, as proposed in CDN-based solution of MAE, was not a component in the original RMM. To allow a fair comparison, we apply

grouping to RMM as well (otherwise, its computational cost would be much worse). This implies that additional measures in the CDN-based solution should also be applied to ensure soundness of group-based planning, e.g., reward adjustment. Let the group size be g and the length of planning horizon be k. The payoff matrix for each RMM agent has a dimension g, the width of each dimension is 5^k, and the total number of cells in the matrix is 5^{kg}.

Each cell is the payoff of the corresponding joint plan mv with k actions for each agent in the group G. Let the sequence of joint actions of agents in G be $mv = (mv^1,...,mv^k)$. Notation mv^i denotes the joint action at the ith step and consists of the ith action of each agent, i.e., $mv^i = \{mv^i_x | x \in G\}$. Let a resultant group trajectory be $t = (c^1, ..., c^k)$, where c^i is the group configuration after joint action mv^i. Configuration c^i consists of the position of each agent, i.e., $c^i = \{ps^i_x | x \in G\}$. The payoff can be computed as the expected group accumulative reward $erw_G(mv) = \sum_{y \in G}(\sum_t (P(t|mv) \sum_{i=1}^k rw_y(c^i)))$, where the second summation is over all possible group trajectories, and $rw_y(c^i)$ is the reward y receives at the group configuration c^i. Since the group configuration c^i is dependent only on the previous configuration c^{i-1} and joint action mv^i, we have $P(t|mv) = \prod_{i=1}^k P(c^i|c^{i-1}, mv^i)$, where $c^0 = null$. Furthermore, since the position of each agent x in group configuration c^i is dependent only on its own action mv^i_x and its own previous position ps^{i-1}_x, we have $P(c^i|c^{i-1}, mv^i) = \prod_{x \in G} P(ps^i_x | ps^{i-1}_x, mv^i_x)$. Combining the above, we have $erw_G(mv) =$

$$\sum_{y \in G} [\sum_t ((\prod_{i=1}^k \prod_{x \in G} P(ps^i_x | ps^{i-1}_x, mv^i_x)) \cdot \sum_{i=1}^k rw_y(c^i))].$$

3.2 Recursive Model Structure

For an agent to use a payoff matrix to plan its actions, it needs the probability of each joint plan, determined by the likelihood of other agents' taking corresponding actions. We identify the key issue for agent B to predict actions of agent A as whether A will move closer to B for cooperation, which is determined by reward distribution around A. Since the reward distribution in the other side of A may be unobservable to B, and RMM agents do not communicate, the above probability must be computed by considering all possible cases of A's neighbourhood. We use recursive modeling as follows:

We characterize the unobservable neighborhood of A by whether it contains high unilateral reward cells. If so, A is more likely to move away from B. Otherwise, A is more likely to move towards B for the benefit of a cooperation. In particular, let nbp^y_x summarize unilateral rewards in neighborhood of agent y that is unobservable to agent x, where $nbp^y_x \in \{$**allLow**,$\neg$**allLow**$\}$. If the unobservable area has at least one high reward, it is labeled $\neg$**allLow**. In general, there are $g-1$ unobservable neighborhoods one per group member, and they form 2^{g-1} possible cases. Each case forms a model at the second level of RMM tree, and is associated with x's belief $P(nbp^1_x, nbp^2_x, ..., nbp^{g-1}_x)$. A two-level RMM tree is used in this work as knowledge at deeper levels cannot be reasonably assumed and deeper models have little effect on performance (Gmytrasiewicz et al., 1998). Fig. 1 shows a RMM tree with $g = 3$ and $k = 2$.

3.3 Bayesian Belief Update

As agents move around, agent x's belief $P(nbp^1_x, nbp^2_x, ..., nbp^{g-1}_x)$ needs to be updated based on observations of other agents' last actions. Let lmv^y_x be the last move of agent y observed by x, where $lmv^y_x \in \{$**towards**, $\neg$**towards**$\}$ and **towards** means that y moved closer to x. To simplify discussion, we assume that $g = 3$, the group consists of agents A, B and C, and $x = B$. Hence, B needs

$$P(nbp^A, nbp^C | lmv^A, lmv^C), \quad (1)$$

we have omitted subscript B to aid readability.

It is difficult, if not impossible, to specify (1) directly as a joint probability of unobserved areas. What can be practically specified is the following as it refers to local dependencies:

$$P(nbp^A | lmv^A) \cdot P(nbp^C | lmv^C). \quad (2)$$

In general, (1) is not equivalent to (2). We show below assumptions needed to obtain (1)

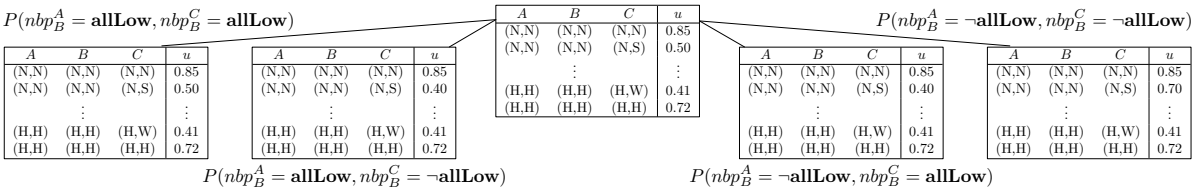


Figure 1: RMM tree of agent B. Payoff matrices shown in table format.

by computing (2). (1) can be rewritten as $P(nbp^A|nbp^C, lmv^A, lmv^C)P(nbp^C|lmv^A, lmv^C)$. If we assume that unobservable neighborhoods of A and C are conditionally independent given their movements, denoted $I(nbp^C, \{lmv^A, lmv^C\}, nbp^A)$, the above probability can be expressed as $P(nbp^A|lmv^A, lmv^C) \cdot P(nbp^C|lmv^A, lmv^C)$. With the additional assumptions $I(nbp^C, lmv^C, lmv^A)$ and $I(nbp^A, lmv^A, lmv^C)$, we obtain (2). Each factor in (2), say, $P(nbp^A|lmv^A)$, can be computed as $\frac{P(lmv^A|nbp^A)P(nbp^A)}{P(lmv^A),}$ where $P(nbp^A)$ is from the last belief update and $P(lmv^A)$ is a normalizing constant. $P(lmv^A|nbp^A)$ can be obtained by reasoning by case based on how rewards in the area between A and B are distributed. Let $m_B^A \in \{\textbf{allLow}, \neg\textbf{allLow}\}$ summarize rewards in this area. We have $P(lmv^A|nbp^A) = \sum_{m^A} P(lmv^A, m^A|nbp^A) = \sum_{m^A} P(lmv^A|m^A, nbp^A)P(m^A|nbp^A)$, where the first factor can be directly specified and the second factor can be estimated based on observed dependence between nearby rewards.

The above relies on the assumptions:

- $I(nbp^A, \{lmv^A, lmv^C\}, nbp^C)$,
- $I(nbp^A, lmv^A, lmv^C)$ and $I(nbp^C, lmv^C, lmv^A)$.

They often do not hold. For the first, when unobservable neighborhoods of A and C overlap, we have $nbp^A = nbp^C$ and the independence no longer holds. The second also fails in this situation since lmv^C is directly dependent on nbp^A. Requirement of these strong assumptions to make (1) computable in practice appears to be a limitation of the RMM framework.

4 Experimental Evaluation

We setup the environment such that the most productive level of cooperation is at two agents. The radius of agent perception and communication is 10 cells. Three types of environments of distinctive natures are simulated. In *Barren* type, each high reward cluster is less than 6×6 in size and is at least 20 cells away from any other high reward cluster. This type is useful to evaluate how well agents can avoid wandering in a low reward area and can migrate to locations with high reward. In *Dense* type, at least every 10×10 square of cells has a high reward cell. In *Path* type, high reward cells form a path and each high reward cell on the path has at least one other high reward cell within a distance of 2 cells.

We set up the CDN-based agent team with group size three, two groups per team, and planning horizon two. Agents, A, B and C, in a group are organized into a chain $A - B - C$. Subnets for A and B are shown in Fig. 2, where design, performance and utility nodes are shown as squares, ovals and diamonds, respectively.

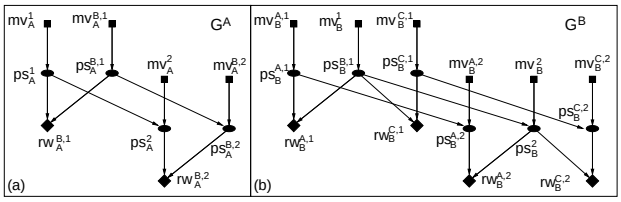


Figure 2: Subnets for group members A and B.

Each movement variable mv_x^i or $mv_x^{y,i}$ generally has 5 possible values denoting the five actions. Each position variable ps_x^1 or $ps_x^{y,1}$ has 5 possible values and each position variable ps_x^2 or $ps_x^{y,2}$ has 13 possible values. The conditional probability table (CPT) associated with a position node encodes uncertain dependency of the position on movement action. The node ps_x^2 is also dependent on the previous location ps_x^1. Utility node $rw_B^{A,i}$ represents rewards that agent

Table 1: Experimental results. Highest means bolded.

	Barren		Dense		Path	
	μ	σ	μ	σ	μ	σ
CDN	**55.84**	4.21	**25.14**	3.27	**20.41**	3.39
GRDU	48.56	0.56	12.32	0.20	12.20	0.15
GRDB	48.64	0.62	18.57	1.10	16.80	2.39
RMM	50.35	5.95	18.50	3.39	18.71	2.79

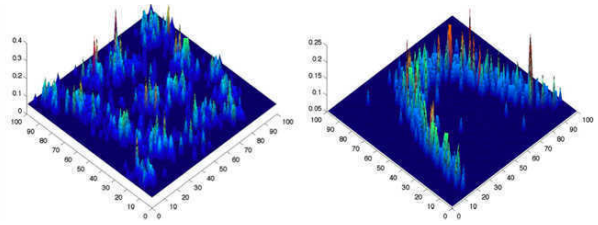
B receives due to cooperation (or lack of) with A. Its associated CPT encodes the reward as a utility distribution. We set up the RMM-based agent team with the same group size, team size and horizon. Both CDN and RMM teams use sophisticated reasoning. To evaluate its benefit, we also implemented two versions of simple greedy agents. One version (GRDU) is based on unilateral reward rwu and selects actions for agent x that maximize $\sum_{i=1}^{k} rwu(ps_x^i)$, where $rwu(ps_x^i)$ is the unilateral reward at the intended position of i'th action. Another version (GRDB) considers bilateral reward rwb as well and maximizes $\sum_{i=1}^{k}(rwu(ps_x^i)+rwb(ps_x^i))$. Each agent acts independently without communication. No group formation is applied as in RMM and CDN. For each version, we set the team size to six.

4.1 Performance Comparison

Tbl. 1 shows experimental performance of each agent team in different environments. For *Barren* type (base value 0.1), each team executes 40 time-steps (80 actions planned) in each run. For *Dense* (Fig. 3(a)) and *Path* (Fig. 3(b)) types (base value 0.05), each team executes 20 time-steps (40 actions planned) in each run. Each team performs 30 runs in each type of environment. The table gives the mean μ and standard deviation σ of the accumulative team reward.

CDN agents outperform other agents. The difference is significant at the 1% significance level when two-tailed t-test is used for all instances except *Path*, where CDN is better than RMM at the 5% significance level.

The *Path* type represents environments where all agents can perform well easily due to an abundance of clues. The *Barren* type represents those where all agents would perform poorly because of the lack of opportunities. The *Dense* type represents those where sound planning would best utilize the existing opportunities. Here the CDN shows the most gain in performance compared with alternatives.



(a) (b)

Figure 3: Environment types. High rewards are shown as taller peaks.

CDN agents outperform greedy agents since they coordinate actions to meet at high bilateral reward cells, whereas greedy agents have no coordination. CDN agents also outperform RMM agents, which can be attributed to two limitations of the latter. Firstly, estimation of neighborhood rewards of other agents through behavior observation and Bayesian update is inaccurate, which hinders effective cooperation. The second limitation is due to existence of multiple optimal joint plans. These joint plans promise the same maximal expected reward, but each agent must choose one in the plan. Without communication, each agent may commit to a different plan such that the resultant joint plan is sub-optimal. Note that this problem cannot be solved by social convention in a LCF as we show in the next section. CDN agents do not suffer from this problem as the interface between agents is composed of movement nodes, which explicitly communicates agent actions.

4.2 On Social Convention

A social convention defines, for each agent, without resorting to communication, the action to take when multiple optimal actions exist. We show that no such convention exists that guarantees collectively optimal actions in MAE. Consider S_1 in Fig. 4, where each cell is labelled with its coordinates, the occupying agent A, B or C, the cooperative per-agent reward b and unilateral reward $u < b$, and no agent can perceive beyond two cells. Let the convention be lexicographical, i.e., $goto(v, z) \succ goto(w, z)$

($\succ$ reads *is-preferred-over*) whenever rewards in cells (v, z) and (w, z) are identical but $v < w$. Hence, B would prefer $goto(2, 0)$ to meet A over $goto(4, 0)$ to meet C, because both actions have the same reward. A would prefer $goto(2, 0)$ to meet B and C would prefer $goto(4, 0)$ to meet B, since $b > u$. The joint action is optimal, though C is unable to meet B.

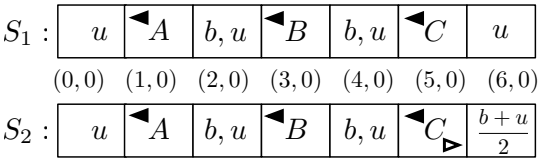


Figure 4: Two scenarios S_1 and S_2 for planning

Next consider S_2 in Fig. 4, where reward for $(6, 0)$ is slightly increased. Preferences of A and B do not change. C still prefers $goto(4, 0)$ to meet B since $b > \frac{b+u}{2}$. The joint action is suboptimal, as C would be better off with $goto(6, 0)$ as shown by the hollow arrow. Without communication, C has no way of knowing the reward at $(2, 0)$ and predicting B's action. Hence, social convention is incapable of coordinating agents with partial observations.

4.3 Efficiency Comparison

CDN, RMM and GRD teams (team size six) use 57, 16, and 0.8 seconds, respectively, for each round of planning. Below, we consider their scalability. Since greedy agents act independently, their efficiency is unaffected by the team size.

With grouping, each payoff matrix in a RMM agent has the size 5^{kg}, where g is the group size and k is the length of horizon. Hence, the space and time complexity of RMM planning grows exponentially with group size.

In comparison, each CDN-based agent group is organized into a hypertree. The necessity of the hypertree organization for exact multiagent probabilistic reasoning is formally established in (Xiang, 2002). The hypertree, together with required agent interfaces, are essential components of tight coupling and ensure optimal decision making in CDN. Hypertree organization also contributes to efficiency. It guarantees that the computational complexity of a CDN-based group is no worse than that of a RMM-based group in the worst case, and is more efficient when the CDN dependency structure is sparse. Fig. 5 shows a possible hypertree organization for MAE with g agents, where the subnet for agent A_2 is similar to that in Fig. 2 (b). The degree of any agent on the hypertree determines the number of agents whose interaction must be modeled and critically determines planning complexity of the agent. As long as this degree is bounded, complexity of computation at each group member does not increase with group size and complexity of planning only grows linearly with group size.

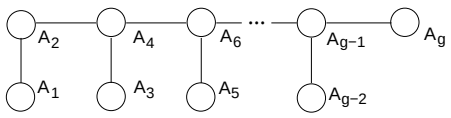


Figure 5: Possible hypertree structure for MAE

5 Discussion

This work is motivated by the disproportional research effort allocated among SVFs, LCFs and TCFs, which forms an odd contrast with the proliferation of distributed/wireless computing, societal emphasis on privacy, and theoretically established advantage of TCFs in utilizing collective knowledge. To improve awareness, we implemented CDN and RMM as representatives of TCFs and LCFs, respectively, in MAE, to allow experimental comparison. The application of RMM to MAE is a novel and nontrivial attempt. The study provided empirical evidence of advantages of CDN over RMM on performance and efficiency. Below we generalize this comparison to other domains and the advantage of CDN over SVFs on privacy.

At the modeling level, RMM and LCFs are limited by the need to model agent interactions without sufficient information. This is evidenced by the need for strong and often invalid assumptions in order to update belief on possible states of team agents (Sec. 3.3). This limitation also applies to communicative LCFs, e.g., (Gmytrasiewicz and Durfee, 2001). Agents in noncommunicative LCFs coordinate by observing other agents' actions. Since messages are speech acts, communicative LCFs are not fun-

damentally different. In contrast, CDN-based agents and TCFs in general do not suffer from this problem as agent interfaces are required to render agent subdomains conditionally independent. RMM is also limited by its matrix-based representation of exponential complexity. This can be remedied by adopting a graphical model in each agent as in MAID (Koller and Milch, 2001), although the above limitation stands.

At the decision making level, RMM and LCFs are limited by having to guess about the states and decisions of other agents based on observations. The inaccuracy in estimation can degrade agent performance through two distinct mechanisms: Firstly through misjudgement of other agents' states, which in turn leads to misjudgement of the optimal joint plan. Secondly, multiple optimal joint plans can degrade agent performance due to independent choice of agents. Social conventions cannot solve this problem as we have shown through a counterexample.

On the other hand, conditional independence rendering interfaces in TCFs convey sufficient states and decisions, resulting in improved coordination and superior performance. Compared to SVFs, an infra-structure exists within each CDN agent to differentiate variables into public and private. Only information on public variables are communicated through an agent interface. Private internal representations and preferences are *not* disclosed. Therefore, TCFs such as CDN provide superior performance, more efficient computation and a higher degree of privacy. Although a cost of communication must be paid (relative to non-communicative LCFs), since the communication is efficient (when the CDN is sparse), the price will be worthwhile for many applications. If communication is noisy/lost the performance degrades gracefully as agents can continue to work in smaller groups (see Sec. 8.9 in (Xiang, 2002)).

Regarding the generality of this work, we draw attention to key features of CDN and RMM. Both are decision-theoretic. RMM is proposed as a general framework for cooperative multiagent decision making. CDN is proposed in the context of collaborative design, but is in fact a general framework, whose applicability to MAE, a domain very different from design, is a clear indication. The generality of CDN and RMM and their common decision-theoretic foundation point to the source of difference in their experimental evaluation, i.e., their difference in agent coupling, and promise that our empirical results in MAE are generalizable.

References

R. Becker, S. Zilberstein, V. Lesser, and C.V. Goldman. 2004. Solving transition independent decentralized Markov decision processes. *J. Artificial Intelligence Research*, 22:423–455.

P.J. Gmytrasiewicz and E.H. Durfee. 2001. Rational communication in multi-agent enviroments. *Auto. Agents and Multi-Agent Systems*, 4(3):233–272.

P.J. Gmytrasiewicz, S. Noh, and T. Kellogg. 1998. Bayesian update of recursive agent models. *User Modeling and User-Adapted Interaction*, 8(1):49–69.

D. Koller and B. Milch. 2001. Multi-Agent Influence Diagrams for Representing and Solving Games. In *Proc. 17th Inter. Joint Conf. on Artificial Intelligence*, pages 1027–1034.

S. Maes, K. Tuyls, and B. Manderick. 2001. Modeling a multi-agent enviroment combining influence diagrams. In M. Mohammadian, editor, *Proc. of IAWTIC*, pages 379–384.

P.J. Modi, W. Shen, M. Tambe, and M. Yokoo. 2005. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180.

A. Petcu and B. Faltings. 2005. A scalable method for multiagent constraint optimization. In *Proc. 19th Inter. Joint Conf. on Artificial Intelligence*, pages 266–271.

J. Shen and V. Lesser. 2006. Communication management using abstraction in distributed Bayesian networks. In *Proc. 5th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 622–629.

Y. Xiang and F. Hanshar, 2007. *Advances in Artificial Intelligence, LNCS*, volume 4509, chapter Planning in Multiagent Expedition with Collaborative Design Networks, pages 526–538. Springer-Verlag.

Y. Xiang, J. Chen, and W.S. Havens. 2005. Optimal design in collaborative design network. In *Proc. 4th Int. Joint Conf. on Auto. Agents and Multiagent Systems*, pages 241–248. ACM.

Y. Xiang. 2002. *Probabilistic Reasoning in Multiagent Systems: A graphical models approach*. Cambridge University Press.

Y. Xiang. 2007. Tractable optimal multiagent collaborative design. In *Proc. IEEE/WIC/ACM Inter. Conf. on Intelligent Agent Technology*, pages 257–260.

Author Index

Agosta, J.M., 1
Antonucci, A., 17
Ammar, S., 9
Arias, M., 25
Aussem, A., 81

Bolt, J. H., 33

Cano, A., 41, 49
Chen, T., 57
Cobb, B., 65
Cordero H., J., 73

Daneshkhah, A.D., 265
de Campos, C. P., 17
de Morais, S.R., 81
de Waal, P.R., 89
Defourny, B., 9
Díez, F.J., 25, 97
Druzdzel, M.J., 1, 185

Elizalde, F., 97

Fernández, A., 105
Flesch, I., 113
François, O.C.H., 121

Gámez, J. A., 129, 137
Gardos, T.R., 1
Gómez-Olmedo, M., 41
Gómez-Villegas, M.A., 145

Hanshar, F., 305

Jensen, F.V., 153, 177, 233, 289

Kontkanen, P., 257
Kwisthout, J., 161

Langseth, H., 169
Leray, P., 9
Lozano, J.A., 249
Lucas, P.J.F., 113
Luque, M., 97, 177

Maaskant, P.P., 185
Madsen, A., 193, 201
Main, P., 145
Masegosa, A., 49
Mateo, J. L., 129, 137
Mendiburu, A., 249
Mononen, T., 209
Moral, S., 41, 49
Myllymäki, P., 209, 257

Nielsen, J.D., 105, 217
Nielsen, T.D., 129, 169, 177
Niepert, M., 225

Ottosen, T.J., 233

Puerta, J.M., 129, 137

Renooij, S., 241
Reyes, A., 97
Roos, T., 257
Rumí, R., 169, 217

Salmerón, A., 105, 169, 217
Santana, R., 249
Savicky, P., 297
Silander, T., 257
Smith, J.Q., 265
Stefanini, F.M., 273
Studený, M., 281
Sucar, L. E., 97
Sun, Y., 17
Susi, R., 145
Søndberg-Jeppesen, N., 289

van der Gaag, L.C., 241
Van Gucht, D., 225
Vomlel, J., 281, 297

Wang, Y., 57
Wehenkel, L., 9

Xiang, Y., 305

Zaffalon, M., 17
Zeng, Y., 73
Zhang, N. L., 57