PGM 2010

Proceedings of the Fifth European Workshop on Probabilistic Graphical Models

Edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola

Helsinki Institute for Information Technology HIIT
HIIT Publications 2010–2
ISBN 978–952–60–3314–3 (electronic)
ISSN 1458–946X

Proceedings of The Fifth European Workshop on Probabilistic Graphical Models

13-15 September, 2010

Helsinki, Finland

Petri Myllymäki, Teemu Roos and Tommi Jaakkola,

editors

Conference Organization

Programme Chairs

Petri Myllymäki
Teemu Roos
Tommi Jaakkola

Organizing Chairs

Petri Myllymäki
Teemu Roos

Programme Committee

Concha Bielza
Wray Buntine
Adnan Darwiche
Luis de Campos
Francisco Diez
Marek Druzdzel
Ad Feelders
Julia Flores
Jose Gámez
Christophe Gonzales
Patrik Hoyer
Juan Huete
Manfred Jaeger
Finn Jensen
Radim Jiroušek
Kristian Kersting
Uffe Kjaerulff
Mikko Koivisto
Petri Kontkanen
Matti Kääriäinen
Helge Langseth
Pedro Larrañaga
Jose Lozano
Peter Lucas
Serafin Moral
Jens Nielsen
Thomas Nielsen
Kristian Olesen
Jose Peña
Jose Puerta
Silja Renooij
David Rios
Antonio Salmerón
Prakash Shenoy
Tomi Silander
Jim Smith
Harald Steck
Milan Studený
Marco Valtorta
Linda van der Gaag
Jiří Vomlel
Marta Vomlelová
Jon Williamson
Yang Xiang
Huizhen Yu
Marco Zaffalon
Nevin Zhang

Additional reviewers

Juan I. Alonso-Barba, Tao Chen, Sander Evers, Arjen Hommersom, Thorsten Ottosen, Yi Wang.

Sponsors

The financial support of University of Helsinki, Helsinki Institute for Information Technology HIIT, Federation of Finnish Learned Societies, Finnish Cultural Foundation (through the Studia Stemmatologica project), Pascal Network of Excellence, and Microsoft Research is gratefully ackowledged.

Preface

The European Workshop on Probabilistic Graphical Models (PGM) is a biennial workshop that brings together researchers interested in all aspects of graphical models for probabilistic reasoning, decision making, and learning. PGM 2010 is the fifth edition of the workshop, and took place in Helsinki, Filnland, in September 13–15, 2010. Previous meetings in the series were held in Cuenca, Spain (PGM 2002), Leiden, Netherlands (PGM 2004), Prague, Czech Republic (PGM 2006), and Hirtshals, Denmark (PGM 2008).

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 57 papers submitted to the workshop. The papers went through a rigorous reviewing process, where each submission was reviewed by at least three members of the PGM2010 programme committee. We would like to thank the 47 programme committee members for their outstanding job during the review process, which is of course crucial for a successful workshop. In addition to the programme committee members, there were also six additional reviewers to whom we are also most grateful.

Of the 57 submissions, 36 were accepted for presentation at the workshop, where each accepted paper was given a slot both in a plenary and in a poster session taking place on the same day as the talk. In addition to the presentation of technical papers, we were very pleased to have three distinguished invited speakers this year: Adnan Darwiche (UCLA), Chris Howe (University of Cambridge) and Thore Graepel (Microsoft Research). The first of the invited talks was organized jointly with a co-located event, the 12th European Conference on Logics in Artificial Intelligence (JELIA 2010).

Petri Myllymäki, Teemu Roos and Tommi Jaakkola
September 2010

Contents

Evaluation Results for a Query-Based Diagnostics Application

John Mark Agosta, john.m.agosta@intel.com
Intel Labs, Santa Clara, CA, USA

Omar Zia Khan and Pascal Poupart {ozkhan,ppoupart}@cs.uwaterloo.ca
University of Waterloo, Waterloo, Ontario, Canada

Abstract

Query-Based Diagnostics refers to the simultaneous building and use of Bayes network diagnostic models, removing the distinction between elicitation and inference phases. In this paper we describe a successful field trial of such a system in manufacturing. The detailed session logs that are collected during use of the system reveal how the model evolved in use, and pose challenging questions on how the models can be adapted to session outcomes.

1 Introduction

Diagnostic models for optimizing fault isolation represented as Bayes networks is a mature field that has seen numerous practical applications. These models have been shown to perform well given one has a comprehension model of the domain to which they apply. In many cases the challenge is to come up with an adequate model: This is the so-called "knowledge elicitation bottleneck." As described in a previous paper (Agosta et al., 2008) we proposed an approach to ease model creation, by combining the elicitation and model-building phase with the model use phase so that the model is improved as it is used. This combination accords well with the expert's view of the process: In the midst of actual problem-solving is the time when the expert is most aware of her mental model, and is best able to express it. We've called this approach *query-based diagnostics,* (QBD) to highlight the dependence of the model on the user's inquiries as they use the model. We report here the field trial results that we had proposed to undertake then. The reader is referred to that paper for the motivation and details of the web-based application which is described therein.

A diagnostic troubleshooting QBD application employing these ideas has been successfully fielded and evaluated by Intel Manufacturing to validate its use in practice. We will refer to it here as "The QBD Application." This paper describes the design and scope of the evaluation trial and the factors that led to its success.

At the start of our involvement with our client, we attempted to implement knowledge-based Bayes network diagnostics for corrective maintenance of factory equipment. Our client was critical of the need to pre-build models by conventional elicitation methods, because engineers and technicians could not be spared to be taken out of their daily activities for model-building, which in their opinion properly belonged under their control. Also repair knowledge is dynamic and would quickly get out of date. The alternative of learning models by statistical methods is not practical because failures are relatively rare by their nature, making available data too sparse. Our response to this quandary was to integrate model creation within the routine work-flow by which they were trained, by this aforementioned approach of mingling model creation and use.

Corrective maintenance refers to unanticipated failure of equipment that takes it out of production. The transition from operating to not operating is the *breakdown*; the transition back is the *repair*. When a factory operation is capacity limited, as opposed to demand limited, equipment breakdowns that constrain

Pp. 1–9 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

capacity incur significant revenue opportunity losses. The purpose of the evaluation trial was to validate that such losses could be substantially avoided by adoption of the QBD Application.

The QBD Application contributes to the Lean Manufacturing philosophy of the firm, specifically by standardizing problem-solving for diagnosis and repair. "Lean" refers to the methods made popular by Toyota for continuous efficiency improvement (Womack, 1990). The justification for the Application as a way to promote knowledge-sharing and the criteria on which it was evaluated both have their roots in "Lean."

The trial was evaluated by the QBD Application's estimated contribution to the overall profitability of the factory. By virtue of running the software in the field we were also able to collect detailed session logs capturing user actions that revealed how the Application performed. After a brief look at the relevant literature in Section 2, we detail the analysis and findings of the evaluation, and the case they make for QBD in Section 3.

The success of the trial leads to several new challenges. One raised in our previous work is the adaptation of the model as sessions generate verified cases. In the course of the trial it also became apparent that user feedback, both active and passive in the sense of expert users following or not following the model's recommended steps can be used to improve the model. We discuss this in Section 4 of the paper.

2 Background

For the developments in Bayes networks for building normative troubleshooting models and the flurry of research done in the 80's and 90's on this topic we refer the reader to (Jensen, 2001), particularly in the bibliography included in the Preface. In a conventional application the model guides the user during a diagnostic session with a ranking of causes and tests. Causes are ranked by their marginal posteriors, conditioned on the evidence offered by the user, and tests are ranked by diagnostic value, often approximated as mutual information, or decrease in entropy of the causes' marginals. The model is run repeatedly, creating a dialog with the user of test suggestions alternating with test execution and the entering of new evidence into the model.

As described in the previous paper, the QBD application extends the concept of a diagnostic session by with editing functionality for active input of causal relations between variables, and simple inferences of causal relations based on passive observation of model building steps all captured by detailed logging of diagnostic sessions. Thus at any point in the session, the actions available to the user are

1. Enter a test result as evidence,
2. Create a new cause or new test node,
3. Add or remove a dependency arc between a cause and test node,
4. Choose a cause on which to perform a repair, ending the session.

In this way we were able to create models without making the Bayes network explicit. The goal is to have the software bootstrap the modeling task, so that, in principle the software could be fielded before any model-building commenced. Not surprisingly the model complexity is limited. The models are relatively sparse bipartite graphs with probabilities set qualitatively; see Figure 2 for an example. Despite their simplicity they suit the situation well, with the right blend of dynamicism, ease of use and comprehensive knowledge-capture.

There is a small but growing literature on applications of diagnostics Bayes networks and their evaluation in use (Pourret et al., 2008). This literature is largely concerned with their accuracy (Przytula et al., 2003). Published cases where the net benefit of an application in use are unknown to the authors, especially of the type we offer here.

3 Evaluation Trial

This section describes the purpose of the trial, the evaluation design, and what the results were.

The trial ran for seven weeks in one factory, with the team responsible for four types of equipment used in semiconductor process fabrication. The evaluation effort was carried out by the team that owns the Lean program.

3.1 Purpose of evaluation

An "ROI" study is required by the firm before it will adopt new software. Although the factory has implemented comprehensive data collection and analysis in all aspects of process and equipment monitoring, there is no system for the capture and use of "soft" user knowledge. In particular, support for maintenance activities extends only to on-line display of reference manuals. The QBD Application was the first trial in Manufacturing where the collection of soft data has been justified by its effect on improving key measures of performance.

3.2 ROI analysis

With the caveat that confidentiality considerations prevent revealing in this paper actual financial cost numbers, we can say that the trial showed substantial improvements in key measures of performance by which the success of the factory is evaluated. Expressed in dollar amounts, the net value of the trial would more than justify the annual budget for the entire lab where the Application was developed!

In the ROI analysis design, the value of the Application was expressed solely by its effect on equipment utilization. Capital utilization and resultant improvement in revenue dwarfed all other cost and benefit terms. Direct labor and material cost and savings, for instance were immaterial in comparison.

3.2.1 Trial and Model Assumptions

Despite the use of the term "ROI", the value analysis consisted of computing change in revenue net of cost, extrapolated to a year.

Assumptions: Three assumptions were made: 1) The equipment to which the model is applied constrains current production capacity. 2) The model would self-populate during the trial. 3) Technicians of various abilities would be involved, some contributing knowledge to the Application, others benefiting from the knowledge that had been entered.

Improvement in Means over Baseline	
MTBF	14%
MTTR	24%

Table 1: Mean values for performance improvements for weeks 14-20 compared to weeks 1-13.

Benefit: Management systems in the factory track Time To Repair (TTR) and Time Between Failure (TBR) by recording the times each machine goes down and comes back up. The change in equipment availability is a function of Mean TTR and Mean TBR. Improvement in factory throughput is estimated from availability of the capacity-limiting equipment step in the production process. Then, knowing weekly production throughput and financial contribution of each unit of product, additional revenue can be estimated. In short, starting with uptime and downtime statistics to estimate improvement in availability, one can estimate

$$\begin{aligned} &increased\_revenue \\ &= availability\_improvement(MTTR, MTBF) \\ &\quad \times production\_volume \\ &\quad \times contribution/unit \end{aligned} \quad (1)$$

Cost: The only direct cost was one additional half day of the planned Lean Manufacturing training. There were no additional R&D or IT infrastructure costs. One might argue that the time spent in sessions with the Application should be counted: From the session logs, we generously estimated that a total of less than two weeks of labor over the entire trial.

3.2.2 Results

We present here quantitative results for one of the four types of equipment involved. The conclusions are similar for each. MTTR and MTBR improved for the pilot period compared to the baseline period. Boxplots for both periods are shown in Figure 1. **The total repair time decreased by 41% while Mean Time to Repair decreased by 24%, as shown in**

Variance	Weeks		Decrease	P value
	1-13	14-20		
Repair Variance	1443.	148.8	0.103	0.0054
Availability Variance	50.51	6.90	0.137	0.011

Table 2: Variance reduction for the trial period compared to the previous period.

Table 1. More impressive is the improvement due to the variance of availability, which decreased by more than a factor of 10, as shown in Table 2, a decrease that passed a conventional test for statistical significance.

Since the QBD Application was an integral part of a new Lean program, it is not possible to tease apart the contribution of the Lean philosophy and the introduction of the Application; the software would not have been used had it not been part of the program, and the Lean workflow would not have been enforced without the software.

Clearly the decrease in repair variance has substantial value for manufacturing performance, on a par with the value of improvements in MTBF and MTTR. Ironically there were no financial analysis techniques available to value this decrease, even though its contribution may be more important than the value attributed to the increase in availability.

3.3 Model-building During Sessions

A benefit of the detailed session logging is the record of when and how model building occurred during diagnostic sessions. In this section we consider the sessions for the three machines for one type of equipment. The session logs captured the entire sequence of user actions: the symptom indicating the breakdown, each cause or test selected, what values tests were set to, and most importantly when causes, tests or dependencies were added to the models.

If we also had had detailed TTR and TBF records for each machine we could have associated sessions with repairs to obtain a more detailed analysis of the relationship between them, and a session-level analysis of effectiveness.

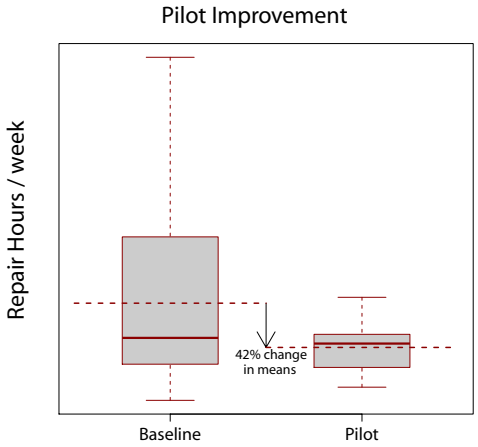


Figure 1: Compared to the previous period, the mean repair hours per week decreased substantially, relieving a production bottleneck and leading to an increase in overall factory availability.

We validated our presumption in the previous paper, that **within a few months models would mature in size, incorporating a useful level of coverage.** Model building occurred continually and extensively during the trial, validating our supposition that intermingling model-building with diagnostic sessions can replace conventional means of model elicitation. During the trial's seven weeks, users ran 157 *sessions,* 54 of which terminated with a repair action. Also we counted the number of diagnostic and model building *actions* in each session; the cumulative counts over all sessions, shown in Figure 3, show the rate of model-building. Our surrogate for diagnostic steps were the actions of setting the value of a test variable. Model building steps (e.g. adding a cause, test or dependency) out-paced diagnostic steps, except for a few short intervals near the end of the trial. We interpret this to imply that the models had not yet reached "saturation" by the trial's end. We would expect that we'd see the rate of testing surpass the rate of additions to the model as models continued to mature.

The arcs shown in the created network, Figure 2, were inferred from the sequence of user

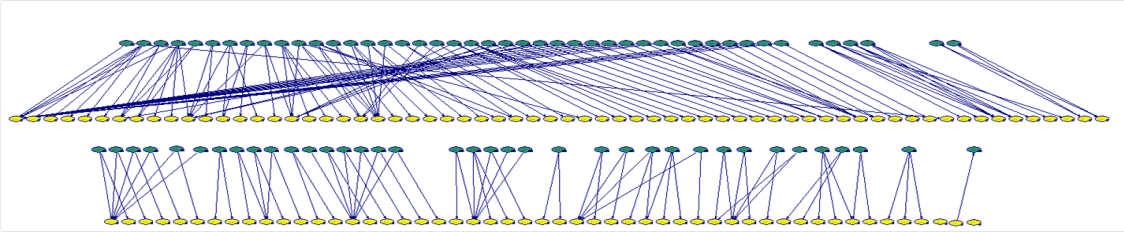


Figure 2: The bipartite network created in use during the pilot contains 115 test and 82 root cause variables, connected by 188 dependencies. Causes appear in the top row, and tests in the bottom.

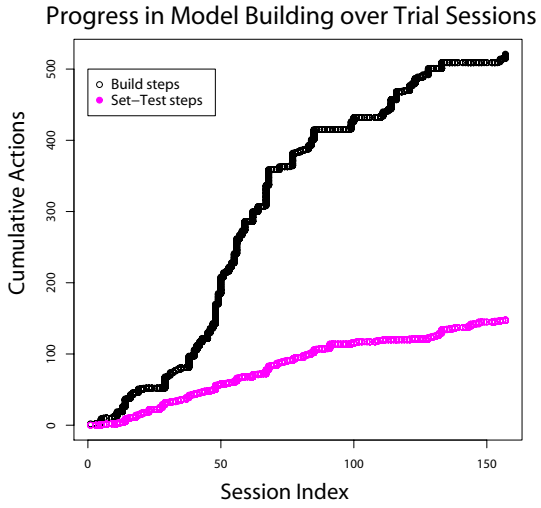


Figure 3: Model building occurred consistently over the course of the 157 sessions during the trial, with a slight decline during the end, and out-pacing the rate of diagnostic actions, except occasionally towards the end.

actions. The result is to generate a singly connected, network of test and cause pairs with all causes linked to one common symptom. All arcs that were not generated in this automatic fashion were added explicitly, using a cause-to-test linking feature in the interface.

4 Adaptation

In this section we present a method where we can make improvements to models by incorporating the results of explicit user feedback from sessions. There are two types of feedback we consider, first the user's ranking of causes, secondly of tests. Both types are represented as cases, serving as constraints to which the model should conform, as explained in the previous paper, where this consistency condition was presented:

Definition 1 (Case Cause-Consistency[1]). A model M^* is *cause-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: $\mathsf{P}\left(C^{(1)} \mid \mathbf{e}^{(j)}, M^*\right) \geq \ldots \geq \mathsf{P}\left(C^{(k)} \mid \mathbf{e}^{(j)}, M^*\right)$.

A case that raised cause consistency would occur if, after completing the diagnosis test sequence, the diagnostician discovers in the course of repair that C^* is the cause of the breakdown, not the C recommended by the model.

Extending the last paper, we consider another consistency condition that follows from the user's behavior in selecting tests in a different order than the diagnostic ranking provided by the model.

Definition 2 (Case Test-Consistency). A model M^* is *test-consistent* with a case j, to level k, if the list of ordered diagnostic test values $\mathsf{MI}\left(T^{(i)} \mid \mathbf{e}^{(j)}\right)$ for tests $T^{(i)}$ given the evidence $\mathbf{e}^{(j)}$ agrees with the case: $\mathsf{MI}\left(T^{(1)} \mid \mathbf{e}^{(j)}, M^*\right) \geq \ldots \geq \mathsf{MI}\left(T^{(k)} \mid \mathbf{e}^{(j)}, M^*\right)$.

For example, if, after observing the vector of evidence $\mathbf{e}$, the diagnostician picks test T^* rather than a test T that is ranked with higher diagnostic value, then the model is inconsistent with the diagnostician's choice. In the QBD Application, diagnosticians cannot indicate an inconsistency in test selections directly, but inconsistencies appear when the highest ranked test computed by the model with the evidence at that point in the session sequence is not the one chosen by the diagnostician. This is can be determined by running the model and comparing

[1] This is the definition from (Agosta et al., 2008)

its recommendations to test actions in the session log.

4.1 Modifying the network

If we believe the cases offered by experts that are inconsistent with the model are correct (and thus the model is giving the wrong recommendation), the model can be improved by modifying it to be consistent with the cases.

Consider the simple case of a four node network, as shown in Figure 4, but with the arc from C_1 to T_2 missing, such that the network is singly connected. The upper layer nodes labeled with C_x represent possible causes. The lower layer nodes labeled T_y represent possible diagnostic tests. Assume the case where the diagnostician ranks T_1 higher than T_2, in contrast to the model's ranking, which is the reverse. Consistency could be restored by either modifying the CPT of T_1, since it arbitrates between the two causes, or, assuming T_1 is a Noisy-OR, symmetric in both causes, exploiting its "Independence of Causal Influences" property using T_2 to change the balance between causes. The first approach is more direct, and seems most natural given its role in relating the two causes.

In the opposite case where the diagnostician ranks T_2 higher than T_1, in contrast to the model's ranking, the same options apply, to change the CPT of either T_1 or T_2 to restore consistency. However it may be most natural to add the arc from C_1 to T_2, so that T_2 is informative of both causes.

Once the network is multiply connected, such as the fully connected version in Figure 4, then intuitions are not so simple about how to modify the network to achieve consistency, and we propose a method that resorts to formulating the problem as a constrained optimization problem. We show in the next section how this might be done, and why the constraints being non-convex make this problem challenging.

4.2 Model Refinement as Optimization

We consider how a test-consistency constraint may be applied to an existing model by revising the parameters of the model such that the model becomes consistent with the case. To elaborate our approach for model refinement, we use the simple diagnostic Bayesian network shown in Figure 4.

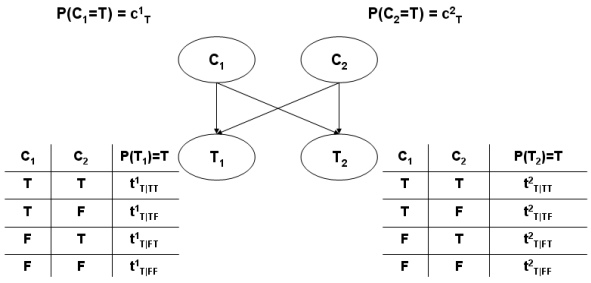


Figure 4: A bipartite diagnostic Bayes network with two causes and two tests.

Assume that both causes and tests are binary variables. The variable $c_i^{(x)}$ indicates the value for $\mathsf{P}(C_x = i)$ whereas the variable $t^y_{m|c_1c_2}$ indicates the value for $\mathsf{P}\left(T_y = m \,|\, C_1 = c_1, C_2 = c_2\right)$. The tables in Figure 4 only show half of the CPTs for these variables for the case where the value of the variable is true.

If a model is inconsistent with a case, it means that the parameters of this Bayesian network *i.e.*, the Conditional Probability Tables (CPTs) are not accurate due to which the ranking of tests is incorrect. Thus, our goal is to refine these parameters (or CPTs). If the model is consistent with a case, we may not need to refine the parameters, however it is still necessary to impose a constraint so that future refinements due to another case do not result in a model that is inconsistent to a previous case.

Consider the case where an expert chooses test T_1 initially without any evidence. We express the diagnostic value as mutual information. Thus the mutual information $\mathsf{MI}(C_1, C_2 \,|\, T_1)$ of the causes C_1, C_2 and test T_1 will be higher than that of the causes C_1, C_2 and test T_2. We can express this information as the following inequality constraint.

$$\mathsf{MI}(C_1, C_2 \,|\, T_1) \geq \mathsf{MI}(C_1, C_2 \,|\, T_2) \tag{2}$$

Writing out the mutual information:

$$\mathsf{MI}(C_1 \ldots C_n \,|\, T_y) = \mathsf{H}(C_1 \ldots C_n) - \sum_{m \in \{T,F\}} t^y_m \mathsf{H}(C_1 \ldots C_n | T_y = m), \tag{3}$$

where $\mathsf{H}(C_1 \dots C_n)$ refers to the joint entropy of the network's causes, computed as follows:

$$\mathsf{H}(C_1 \dots C_n) = - \sum_{c_1, c_2 \in \{T,F\}} \mathsf{P}(c_1, c_2) \log_2 (c_1, c_2). \tag{4}$$

Using Equations 3 and 4, we can rewrite the constraint in Equation 2 as follows.

$$\begin{aligned} 0 \geq & - \sum_{t_1, c_1, c_2 \in \{T,F\}} \mathsf{P}(t_1) \mathsf{P}\left(c_1, c_2 \,|\, t_1\right) \log_2 \mathsf{P}\left(c_1, c_2 \,|\, t_1\right) \\ & + \sum_{t_2, c_1, c_2 \in \{T,F\}} \mathsf{P}(t_2) \mathsf{P}\left(c_1, c_2 \,|\, t_2\right) \log_2 \mathsf{P}\left(c_1, c_2 \,|\, t_2\right) \end{aligned} \tag{5}$$

The constraint shown in Equation 5 is non-linear. This is evident if we use Bayes' theorem and rewrite the constraint only using the parameters of the Bayes network as shown here:

$$\begin{aligned} 0 \geq & \sum_{i,j \in \{T,F\}} c_i^{(1)} c_j^{(2)} \sum_{y=1}^{2} \sum_{m \in \{T,F\}} t_{m|i,j}^{(y)} \times \\ & \left[\log_2 t_{m|i,j}^{(y)} + \log_2 c_i^{(1)} + \log_2 c_j^{(2)} - \log_2 k_m^{(y)} \right] \end{aligned} \tag{6}$$

In the above equation, $k_m^{(y)}$ are normalization constants. They can be evaluated as follows.

$$k_m^{(y)} = \sum_{i,j \in \{T,F\}} t_{m|i,j}^{(y)} c_i^{(1)} c_j^{(2)}$$

The CPTs for the causes are the priors over those causes. This information can be learned using historical data as it only relates to the frequency with which a cause occurs. Thus, we can exclude this from the set of the parameters to be learned or refined and restrict ourselves to the CPTs of the tests, which are of the form $t_{m|i,j}^{(y)}$ in Equation 6.

The above constraint is the simplest possible version of a network, where there are only two causes, two nodes and no evidence available. For more complex networks, the constraints will be more complicated.

Most significantly, this constraint is not convex. It is well-known that entropy is concave, while mutual information is neither concave nor convex (Cover and Thomas, 1991). Since our constraint is a difference of mutual information it follows that it is a difference of two non-convex functions. There has been previous work on using EM algorithm with constraints to learn the parameters of a Bayes network (Niculescu et al., 2006). However, the constraints considered in those settings are linear, and thus convex.

This problem is also different as unlike traditional parameter learning problems, we do not have a lot of data to learn from. Thus, the traditional objective function of maximizing the data log likelihood may also not be useful as it could lead to over-fitting. Finally, we are already provided an existing model which needs to be refined rather than a new model learned from scratch. An alternate approach is to minimize distance from the original model using a measure such as an L_p-norm. However, this may not lead to a robust approach since when the model is not consistent, the distance measure will force the refined model to be at a corner point of the new feasible region. Instead, we are interested in an objective function that forces the parameters of the refined model to lie in the interior of the new feasible region, as far as possible from all the constraint surfaces. In this manner, the refined model is more likely to be consistent with any future constraints. Currently we are investigating the use of KL-divergence as a possible objective function that can help us achieve this goal.

5 Conclusion

The results presented in this paper show overwhelming benefits from employing an implementation of a QBD Application; benefits that outweighed costs by orders of magnitude, and made a material difference in the factory's estimated financial performance. One may wonder what this is attributable to, given the simplicity of the models and their lack of maturity, being created in the course of the trial. The answer is that the success of the trial relied on success in all three areas, financial, organizational and human interface. The lessons learned from this trial are

several:

1. Diagnostic applications have a natural application in improving equipment efficiency. In capital-intensive companies, improving capital efficiency justifies such applications since these savings dwarf those in other areas such as direct costs of labor and materials.

2. Lean manufacturing principles pave the way for introducing technology to improve efficiency, by making it clear to the organization where the technology fits. Management in the organization views the application as a way to capture and share "tribal" knowledge for Lean problem solving.

3. A new technology has to make the job easier for the person who uses it, or else it will not be adopted. Here the users comments were varied, ranging from "no tool (i.e. computer application) has as comfortable a user interface as QBD"; to critical: "System design still needs improvement to simplify usage" and "(It) Would be best if system was integrated into existing tools."

It is well known that working in a domain is facilitated by agreeing on a common, familiar vocabulary with the domain expert. Analogously, adopting a standard work-flow from the way that the client believes the diagnostic task should be performed, as the application workflow is also key to the application's acceptance. Knowledge elicitation methods recognize the importance of process and task analysis (Schreiber et al., 2000). Ours is a stronger statement, that the process steps incorporated in the application adopt the normative model that the organization uses.

In addition to the validation of the application, the trial identified areas for future work. The plan for the current software is to improve the interface and integration into existing software systems, and prove these changes in additional trials. Also the trial revealed other places where user actions may be used as input for the automated learning methods we are developing to improve the model, for instance Test-Consistency. Furthermore we are also considering the applicability of QBD to other diagnostic problem solving tasks, and to extend the method with test and repair costs and with replacement and repair actions, in a dynamic model.

Acknowledgments

It is our pleasure to give credit to Marek Druzdzel, Thomas Gardos, Matthias Giessler, Dan Peters, and Bryan Pollard, who in different ways contributed to and made this project possible. The QBD Application was built using the SMILE library.

References

John Mark Agosta, Tom R. Gardos, and Marek J. Druzdzel. 2008. Query-based diagnostics. In *Probabilistic Graphical Models*, September.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory.* John Wiley & Sons, Inc.

Finn V. Jensen. 2001. *Bayesian Networks and Decision Graphs.* Springer Verlag.

Radu Stefan Niculescu, Tom M. Mitchell, and R. Bharat Rao. 2006. Bayesian network learning with parameter constraints. *Journal of Machine Learning Research*, 7:1357–1383.

Olivier Pourret, Patrick Na'im, and Bruce Marcot, editors. 2008. *Bayesian Networks: A Practical Guide to Applications.* John Wiley & Sons, Inc.

K. Woytek Przytula, Denver Dash, and Don Thompson. 2003. Evaluation of Bayesian networks used for diagnostics. In *IEEE Proceedings Aerospace Conference.*

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.

James P. Womack. 1990. *The Machine That Changed the World.* Simon & Schuster.

Lifted Belief Propagation: Pairwise Marginals and Beyond

Babak Ahmadi and Kristian Kersting and Fabian Hadiji
Knowledge Dicovery Department, Fraunhofer IAIS
53754 Sankt Augustin, Germany
firstname.lastname@iais.fraunhofer.de

Abstract

Lifted belief propagation (LBP) can be extremely fast at computing approximate marginal probability distributions over single variables and neighboring ones in the underlying graphical model. It does, however, not prescribe a way to compute joint distributions over pairs, triples or k-tuples of distant random variables. In this paper, we present an algorithm, called *conditioned* LBP, for approximating these distributions. Essentially, we select variables one at a time for conditioning, running lifted belief propagation after each selection. This naive solution, however, recomputes the lifted network in each step from scratch, therefore often canceling the benefits of lifted inference. We show how to avoid this by efficiently computing the lifted network for each conditioning directly from the one already known for the single node marginals. Our experimental results validate that significant efficiency gains are possible and illustrate the potential for second-order parameter estimation of Markov logic networks.

1 Introduction

There has been much recent interest in methods for performing lifted probabilistic inference, handling whole sets of indistinguishable objects together, see e.g. (Milch et al., 2008; Sen et al., 2009) and references in there. Most of these lifted inference approaches are extremely complex, so far do not easily scale to realistic domains and hence have only been applied to rather small artificial problems. A remarkable exception are lifted versions of belief propagation (Singla and Domingos, 2008; Kersting et al., 2009). They grouped together random variables that have identical computation trees but now run a modified belief propagation (BP) on the resulting lifted, i.e., clustered network. Being instances of BP, they can be extremely fast at computing approximate marginal probability distributions over single variable nodes and neighboring ones in the underlying graphical model. Above all, they naturally scale to realistic domain sizes. Despite their success, however, lifted BP approaches do not provide a prescription to compute joint probabilities over pairs of non-neighboring variables in the graph. When the underlying graphical model is a tree, there is a single chain connecting any two nodes, and dynamic programming techniques might be developed for efficiently integrating out the internal variables. When cycles exist, however, it is not clear what approximate procedure is appropriate. The situation is even more frustrating when computing marginals over triples or k-tuples of distant nodes. As for the non-lifted case, sophisticated exact lifted inference algorithms are only tractable on rather small models and do not scale to realistic domain sizes. It is precisely this problem that we are addressing in this paper, that is we are interested in approximate lifted inference algorithms based on the conditioning idea that scale to realistic domain sizes. Specifically, we present *conditioned* LBP (CLBP), a scalable lifted inference algorithm for approximate inference based on conditioning. Essentially, we select variables one at a time for conditioning and run lifted belief propagation after each selection. This naive solution, however, recomputes the lifted network in each step from scratch, therefore often

Pp. 9–17 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

canceling the benefits of lifted inference. We show how to avoid this by efficiently computing the lifted network for each conditioning directly from the one already known for the single node marginals. There has been some prior work for related problems. Delcher *et al.* (1996) propose a data structure that allows efficient queries when new evidence is incorporated in singly connected Bayesian networks and Acar *et al.* (2008) present an algorithm to adapt the model to structural changes using an extension of Rake-and-Compress Trees. The only lifted inference approach we are aware of is the work by Nath and Domingos (2010) that was independently developed in parallel. The authors essentially simulate their lifting procedure for a set of changed variables, obtaining the adapted lifted network.

We also consider the problem of determining the best variable to condition on in each iteration to stay maximally lifted over all iterations and propose a simple heuristic. Our experimental evaluation including experiments on second-order parameter estimation for Markov logic networks (Richardson and Domingos, 2006) shows that significant efficiency gains are obtainable compared to naively running (lifted) BP in each iteration. CLBP may also have future applications in more advanced relational learning tasks such as active learning.

We proceed as follows. We start off by briefly reviewing LBP. Then, we introduce CLBP, prove its soundness, and touch upon the problem of determining the best variable to condition on at each level of recursion. Before concluding, we present the results of our experimental evaluation.

2 Lifted Belief Propagation

Let $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ be a set of n discrete-valued random variables each having d states, and let x_i represent the possible realizations of random variable X_i. Graphical models compactly represent a joint distribution over $\mathbf{X}$ as a product of factors (Pearl, 1991), i.e.,

$$P(\mathbf{X} = \mathbf{x}) = Z^{-1} \prod_k f_k(\mathbf{x}_k) \ .$$

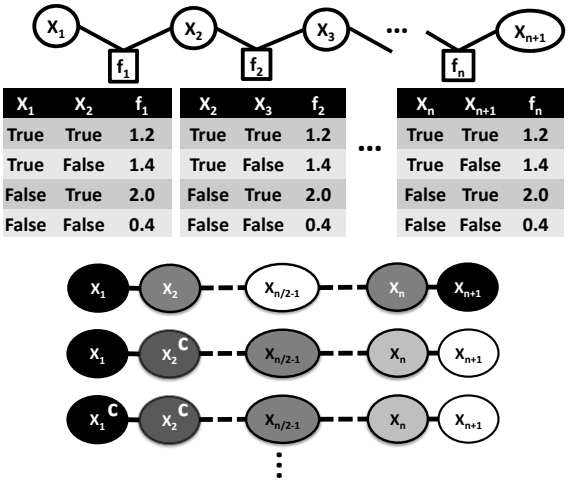


Figure 1: (**Top**) An example for a factor graph — a chain graph model with $n+1$ nodes — with associated potentials. Circles denote variables, squares denote factors. (**Bottom**) Supernodes, indicated by the shades of the original nodes, produced by repeatedly clamping nodes, indicated by "c",on a chain graph model with $n+1$ nodes. Factors have been omitted. The conditioning order is $\pi = \{2, 1, 3, 4, \ldots, n-2, n-1, n+1, n\}$. After clamping X_2 all subsequent LBP runs work on the fully grounded network.

Each factor f_k is a non-negative function of a subset of the variables $\mathbf{x}_k$, and Z is a normalization constant. If $P(\mathbf{X} = \mathbf{x}) > 0$ for all joint configurations $\mathbf{x}$, the distribution can be equivalently represented as a log-linear model: $P(\mathbf{X} = \mathbf{x}) = Z^{-1} \exp\left[\sum_i w_i \cdot g_i(\mathbf{x})\right]$, where the features $g_i(x)$ are arbitrary functions of (a subset of) the configuration $\mathbf{x}$. Each graphical model can be represented as a factor graph. A factor graph, cf. Fig 1 **(top)**, is a bipartite graph that expresses the factorization structure of the joint distribution. It has a variable node (denoted as a circle) for each variable X_i, a factor node (denoted as a square) for each f_k, with an edge connecting variable node i to factor node k if and only if X_i is an argument of f_k. We assume one factor $f_i(\mathbf{x}) = \exp\left[w_i \cdot g_i(\mathbf{x})\right]$ per feature $g_i(\mathbf{x})$.

An important (#P-complete) inference task is to compute the conditional probability of variables given the values of some others, the evidence, by summing out the remaining variables.

The belief propagation (BP) algorithm is an efficient way to solve this problem that is exact when the factor graph is a tree, but only approximate when the factor graph has cycles. Although this loopy BP has no guarantees of convergence or of giving the correct result, in practice it often does, and can be much more efficient than other methods. BP can be elegantly described in terms of sending messages within a factor graph. The message from a variable X to a factor f is

$$\mu_{X\to f}(x) = \prod_{h\in \mathrm{nb}(X)\setminus\{f\}} \mu_{h\to X}(x)$$

where $\mathrm{nb}(X)$ is the set of factors X appears in. The message from a factor to a variable is

$$\mu_{f\to X}(x) = \sum_{\neg\{X\}} \left(f(\mathbf{x}) \prod_{Y\in \mathrm{nb}(f)\setminus\{X\}} \mu_{Y\to f}(y) \right)$$

where $\mathrm{nb}(f)$ are the arguments of f, and the sum is over all of these except X, denoted as $\neg\{X\}$. The messages are usually initialized to 1, and the unnormalized belief of each variable X_i can be computed from the equation

$$b_i(x_i) = \prod_{f\in \mathrm{nb}(X_i)} \mu_{f\to X_i}(x_i)\ .$$

Evidence is incorporated by setting $f(\mathbf{x}) = 0$ for states $\mathbf{x}$ that are incompatible with it. Different schedules may be used for message-passing.

Although already quite efficient, many graphical models produce factor graphs with a lot of symmetries not reflected in the graphical structure. Consider the factor graph in Fig. 1(**top**). The associated potentials are identical. Lifted BP (LBP) can make use of this fact. It essentially performs two steps: Given a factor graph G, it first computes a compressed factor graph $\mathfrak{G}$ and then runs a modified BP on $\mathfrak{G}$. We use fraktur letters such as $\mathfrak{G}$, $\mathfrak{X}$, and $\mathfrak{f}$ to denote the lifted, i.e., compressed graphs, nodes, and factors. For the present paper, only the first step is important, which we will now briefly review.

Step 1 of LBP — Lifting by Color-Passing (CP): Let G be a given factor graph with variable and factor nodes. Initially, all variable nodes fall into $d+1$ groups (one or more of these may be empty) — known states $s_1, \ldots, s_d$, and *unknown* — represented by colors. All factor nodes with the same associated potentials also fall into one group represented by a shade. Now, each variable node sends a message to its neighboring factor nodes saying "I am of color C". A factor node sorts the incoming colors into a vector according to the order the variables appear in its arguments. The last entry of the vector is the factor node's own color. This color signature is sent back to the neighboring variables nodes, essentially saying "You have communicated with these kinds of nodes". The variable nodes stack the incoming signatures together and, hence, form unique signatures of their one-step message history. Variable nodes with the same stacked signatures are grouped together, and a new color is assigned to each group. The factors are grouped in a similar fashion based on the incoming color signatures of neighboring nodes. This CP process is iterated until no new colors are created anymore. As the effect of the evidence propagates through the factor graph, more groups are created. The final lifted graph $\mathfrak{G}$ is constructed by grouping nodes (factors) with the same color (signatures) into *supernodes* (*superfactors*). Supernodes (superfactors) are sets of nodes (factors) that send and receive the same messages at each step of carrying out BP on G and form a partition of the nodes in G. On this lifted network, LBP runs an efficient modified BP (MBP). We refer to (Singla and Domingos, 2008; Kersting et al., 2009) for details.

3 Lifted Conditioning

We are often faced with the problem of repeatedly answering slightly modified queries on the same network. Consider e.g. computing a joint distribution $P(X_1, X_2, \ldots, X_k)$ using LBP. A simple method is the following *conditioning* procedure that we call *conditioned* LBP (CLBP). Let π define a *conditioning order* on the nodes, i.e., a permutation on the set $\{1, 2, \ldots, k\}$ and its i-th element be denoted as $\pi(i)$. The simplest one is $\pi(i) = i$. Now, we select variables one at a time for conditioning, running LBP after each selection, and combine the resulting marginals. More precisely,

1. Run LBP to compute the prior distribution $P(X_{\pi(1)})$.

2. Clamp $X_{\pi(1)}$ to a specific state $x_{\pi(1)}$. Run LBP to compute the conditional distribution $P(X_{\pi(2)}|x_{\pi(1)})$.

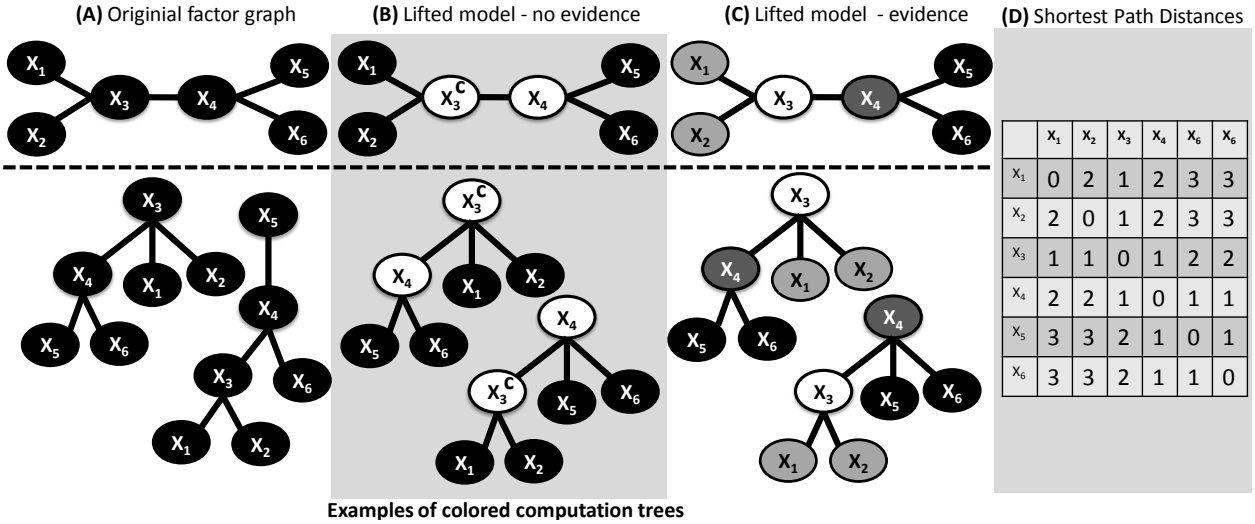
3. Do this for all states of $X_{\pi(1)}$ to obtain all conditional distributions $P(X_{\pi(2)}|X_{\pi(1)})$. The joint distribution is now $P(X_{\pi(2)}, X_{\pi(1)}) = P(X_{\pi(2)}|X_{\pi(1)}) \cdot P(X_{\pi(1)})$.

By iterating steps 2) and 3) and employing the chain rule we have $P(X_1, \ldots, X_k) = P(X_{\pi(1)}, \ldots, X_{\pi(k)}) = \prod_{i=1}^{k} P(X_{\pi(i)}|X_{\pi(i-1)}, \ldots, X_{\pi(1)})$. CLBP is simple and even exact for tree-structured models. Indeed, it is common to apply (L)BP to graphs with cycles as well. In this case the beliefs will in general not equal the true marginals, but often provide good approximations in practice. Moreover, Welling and Teh (2003) report that conditioning performs surprisingly well in terms of accuracy for estimating the covariance[1]. In the lifted case, however, the naive solution of repeatedly calling LBP may perform poorly in terms of running time. We are repeatedly answering slightly modified queries on the same graph. Because LBP generally lacks the opportunity of adaptively changing the lifted graph and using the updated lifted graph for efficient inference, it is doomed to lift the original model in each iteration again from scratch. Each CP run scales $\mathcal{O}(n \cdot m)$ where n is the number of nodes and m is the length of the longest path without loop. Hence, CLBP essentially spends $\mathcal{O}(k \cdot n \cdot m)$ time just on lifting. Moreover, in contrast to the propositional case, the conditioning order has an effect on the sizes of the lifted networks produced and, hence, the running time of MBP. It may even cancel out the benefit of lifted inference. Reconsider our chain example[2] from Fig. 1. Fig. 1(**bottom**) sketches the lifted networks produced over time when using the conditioning order $\pi = \{2, 1, 3, 4, \ldots, n-2, n-1, n+1, n\}$. As one can see, clamping X_2 dooms all subsequent iterations to run MBP on the fully grounded network, canceling the benefits of lifted inference. In contrast, the order $\pi = \{1, n+1, 2, n, \ldots, n/2-1\}$ produces lifted and fully grounded networks alternatingly, the best we can achieve for chain models. We now address both issues.

Shortest-Paths Lifting: Consider the situation depicted in Fig. 2. Given the network in **(A)** and the prior lifted network, i.e., the lifted network when no evidence has been set **(B)**, we want to compute $P(X|x_3)$ as shown in **(C)**. To do so, it is useful to describe BP in terms of its *computation tree* (CT), see e.g. (Ihler et al., 2005). The CT is the unrolling of the (loopy) graph structure where each level i corresponds to the i-th iteration of message passing. Similarly we can view CP, i.e., the lifting procedure as a *colored computation tree* (CCT). More precisely, one considers for every node X the computation tree rooted in X but now each node in the tree is colored according to the nodes' initial colors, cf. Fig. 2**(bottom)**. Each CCT encodes the root nodes' local communication patterns that show all the colored paths along which node X communicates in the network. Consequently, CP groups nodes with respect to their CCTs: nodes having the same set of rooted paths of colors (node and factor names neglected) are clustered together. For instance, Fig. 2**(A)** shows the CCTs for X_3 and X_5. Because their set of paths are different, X_3 and X_5 are clustered into different supernodes as shown in Fig. 2**(B)**. The prior lifted network can be encoded as the vector $l = (0, 0, 1, 1, 0, 0)$ of node colors. Now, when we clamp a node, say X_3, to a value x_3, we change the communication pattern of every node having a path to X. Specifically, we change X_3's (and only X_3's) color in all CCTs X_3 is involved. This is illustrated in Fig. 2**(B)**. For the prior lifted network, the dark and light nodes in Fig. 2**(B)** exhibit the

[1]The symmetrized estimate of the covariance matrix is typically not positive semi-definite and marginals computed from the joint distributions are often inconsistent with each other.

[2]When the graph is a chain or a tree there is a single chain connecting any two nodes and LBP together with dynamic programming can be used to efficiently integrate out the internal variables. When cycles exist, however, it is unclear what approximate procedure is appropriate.



	X_1	X_2	X_3	X_4	X_5	X_6
X_1	0	2	1	2	3	3
X_2	2	0	1	2	3	3
X_3	1	1	0	1	2	2
X_4	2	2	1	0	1	1
X_5	3	3	2	1	0	1
X_6	3	3	2	1	1	0

Figure 2: **(A):** Original factor graph. **(B):** Prior lifted network, i.e., lifted factor graph with no evidence. **(C):** Lifted factor graph when X_3 is set to some evidence. Factor graphs are shown **(top)** with corresponding colored computation trees **(bottom)**. For the sake of simplicity, we assume identical factors (omitted here). Ovals denote variables/nodes. The shades in **(B)** and **(C)** encode the supernodes. **(D):** Shortest-path distances of the nodes. The i-th row will be denoted d_i.

same communication pattern in the network. Consequently, X_3 appears at the same positions in all corresponding CCTs. When we now incorporate evidence on node X_3, we change its color in all CCTs as indicated by the "c" in Figs. 2**(B)** and **(C)**. This effects nodes X_1 and X_2 differently than X_4 respectively X_5 and X_6 for two reasons: (1) they have different communication patterns as they belong to different supernodes in the prior network; more importantly, (2) they have different paths connecting them to X_3 in their CCTs. The shortest path is the shortest sequence of factor colors connecting two nodes. Since we are not interested in the paths but whether the paths are identical or not, these sets might as well be represented as colors. Note that in Fig. 2 we assume identical factors for simplicity. Thus in this case path colors reduce to distances. In the general case, however, we compare the paths, i.e. the sequence of factor colors.

We only have to consider the vector d_3 of shortest-paths distances to X_3, cf. Fig. 2**(D)**, and refine the initial supernodes correspondingly. Recall that the prior lifted network can be encoded as the vector $l = (0, 0, 1, 1, 0, 0)$ of node colors. This is equivalent to (1) $l \oplus d_3$, the element-wise concatenation of two vectors, and (2) viewing each resulting number as a new color. $(0, 0, 1, 1, 0, 0) \oplus (1, 1, 0, 1, 2, 2) =_{(1)} (01, 01, 10, 11, 02, 02) =_{(2)} (0, 0, 1, 2, 3, 3)$, the lifted network for $P(X|x_3)$ as shown in Fig. 2**(C)**. Thus, we can directly update the prior lifted network in linear time without taking the detour through running CP on the ground network. Now, let us compute the lifted network for $P(X|x_4, x_3)$. Essentially, we proceed as before: compute $l \oplus (d_3 \oplus d_4)$. However, the resulting network might be suboptimal. It assumes $x_3 \neq x_4$ and, hence, X_3 and X_4 cannot be in the same supernode. For $x_4 = x_3$, they could be placed in the same supernode, if they are in the same supernode in the prior network. This can be checked by $d_3 \odot d_4$, the element-wise sort of two vectors. In our case, this yields $l \oplus (d_3 \odot d_4) = l \oplus l = l$: the prior lifted network. In general, we compute $l \oplus (\bigoplus_s (\bigoplus_v d_{s,v}))$ where $d_{s,v} = \bigodot_{i \in s: x_i = v} d_i$, s and v are the supernodes and the truth value respectively. For an arbitrary network, however, the shortest paths might be identical although the nodes have to be split, i.e. they differ in a longer path, or in other words, the shortest paths of other nodes to the evidence node are

different. Consequently we iteratively apply the shortest paths lifting. Let SN_S denote the supernodes given the set S as evidence. By applying the shortest path procedure we compute $SN_{\{X_1\}}$ from $SN_\emptyset$. This step might cause initial supernodes to be split into newly formed supernodes. To incorporate these changes in the network structure the shortest paths lifting procedure has to be iteratively applied. Thus in the next step we compute $SN_{\{X_1\}\cup\Gamma_{X_1}}$ from $SN_{\{X_1\}}$, where Γ_{X_1} denotes the changed supernodes of the previous step. This procedure is iteratively applied until no new supernodes are created. This essentially sketches the proof of the following theorem.

Theorem 1. *If the shortest-path colors among all nodes and the prior lifted network are given, computing the lifted network for $P(X|X_i,\ldots,X_1)$, $i>0$, takes $\mathcal{O}(i\cdot n\cdot s)$, where n is the number of nodes, s is the number of supernodes. Running MBP produces the same results as running BP on the original model.*

Proof. For a Graph $G=(V,E)$, when we set new evidence for a node $X\in V$ then for all nodes within the network the color of node X in the $CCTs$ is changed. If two nodes $Y_1,\ Y_2\in V$ were initially clustered together (denoted as $sn_0(Y_1)=sn_0(Y_2)$), i.e. they belong to the same supernode, they have to be split if the $CCTs$ differ. Now we have to consider two cases: If the difference in the $CCTs$ is in the shortest path connecting X with Y_1 and Y_2, respectively, then shortest-path lifting directly provides the new clustering. If the coloring along the shortest paths is identical the nodes' $CCTs$ might change in a longer path. Since $sn_0(Y_1)=sn_0(Y_2)$ there exists a mapping between the paths of the respective $CCTs$. In particular $\exists Z_1,Z_2$, s.t. $sn_0(Z_1)=sn_0(Z_2)$ from a different supernode, i.e. $sn_0(Z_i)\neq sn_0(Y_i)$, and $Y_1,\ldots,\underbrace{Z_1,\ldots,X}_{\Delta_1}\in CCT(Y_1)$, $Y_1,\ldots,\underbrace{Z_2,\ldots,X}_{\Delta_2}\in CCT(Y_2)$ and $\Delta_1\in CCT(Z_1)\neq\Delta_2\in CCT(Z_2)$ are the respective shortest paths for Z_1 and Z_2. Thus, by iteratively applying shortest-path lifting as explained above, the evidence propagates through and we obtain the new clustering. □

On Finding a Conditioning Order: Clearly, CLBP will be most efficient for estimating the probability of a joint state when it produces the smallest lifted networks. This calls for the task of finding the most efficient[3] conditioning order. Here, we provide a generically applicable strategy based on the nodes' shortest-path colors to all other nodes. That is, in each conditioning iteration, we add that node having the smallest number of unique paths to all other nodes and, if possible, is a member of a supernode of one of the already clamped nodes. Intuitively, we select nodes that are expected to create the smallest number of splits of existing supernodes in each iteration. Therefore, we call it *min-split*. Although this increases the running time — each conditioning iteration now has an additional $\mathcal{O}(n^2)$ step — our experiments show that there are important cases such as computing pairwise joint marginals where the efficiency gains achievable due to a better lifting can compensate this overhead.

4 Experimental Evaluation

Our intention here is to illustrate the performance of CLBP compared to naively running LBP and BP. We implemented CLBP and its variants in Python and using libDAI library (Mooij, 2009) and evaluated the algorithms on a number of Markov logic networks.

In our first experiment, we compared CLBP to naively running LBP, i.e. lifting the network each time from scratch, and BP for computing pairwise probabilities. We generated the "Friends-and-Smokers" Markov logic network (Singla and Domingos, 2008) with 2, 5, 10, 15, 20, and 25 people, resulting in networks ranging from 8 to 675 nodes. The shortest-path lifting clearly pays out in terms of the total messages sent (including CP and shortest-path mes-

[3] This question is different from the more common question of finding highly accurate orders. The latter question is an active research area already for the ground case see e.g. (Eaton and Ghahramani, 2009), and is also related to the difficult question of convergent BP variants, see e.g. (Mooij et al., 2007).

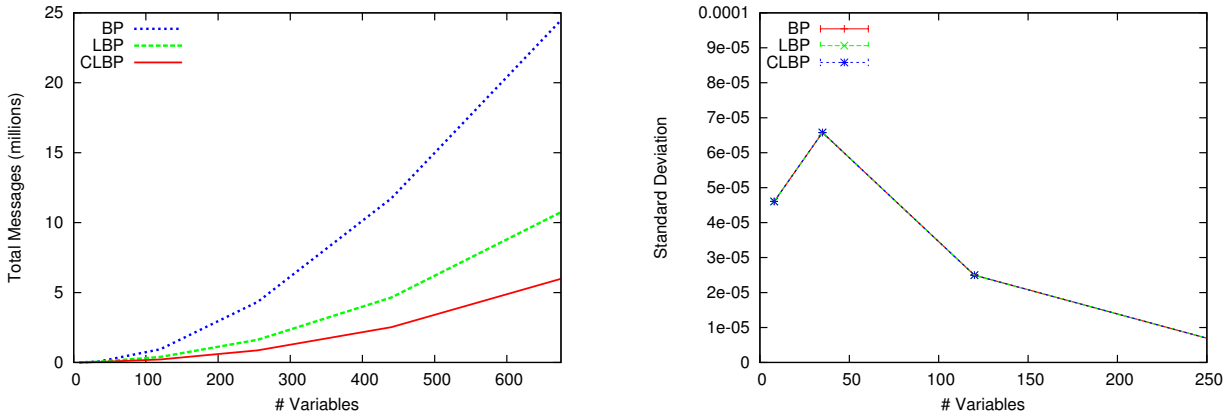


Figure 3: Pairwise Probability Estimates: **(Left)** Comparison of the total number of messages sent for BP, Lifted BP and "min-split" order CLBP for "Friends-and-Smokers" MLNs (including clustering messages for LBP and CLBP). **(Right)** The Standard Deviation of the error compared to the exact solution computed using the Junction Tree Algorithm.

sages) as shown in Fig. 3 (**left**). Moreover, the accuracy estimates are surprisingly good and confirm Welling and Teh (2003); Fig. 3 (**right**) shows the Standard Deviation of the difference compared to the exact solution computed using the Junction Tree (JT). The maximal error we got was below 10^{-4}. Note, however, that running JT with more than 20 persons was impossible due to memory and time restrictions.
In our second experiment we investigated CLBP for computing joint marginals. For the "Friends-and-Smokers" MLN with 20 people we randomly chose 1, 2, ..., 10 "cancer" and "friends" nodes as query nodes. The joint state was randomly chosen. The results are averaged over 10 runs. Fig. 4 shows the cumulative number of messages (including CP messages). "Min-split" is indeed better. By chosing the order following our heuristic the cumulative number of supernodes and in turn messages is reduced compared to a random elimination order.

Finally, we learnt paramters for the "Friends-and-Smokers" MLN with 10 perople, maximizing the conditional marginal log-likelihood (CMLL). Therefore we sampled 5 data cases from the joint distribution. We compared conjugate gradient (CG) optimization using Polak-Ribiere with Newton conjugate gradient (NCG) optimization using the covariance matrix of MLN clauses computed using CLBP. The gradient was computed as described in (Richardson and Domingos, 2006) but normalized by the number of groundings of each clause. The results summarized in Fig. 5 confirm that information about dependencies among clauses is indeed useful: the second order method exhibits faster convergence.

5 Conclusion

We presented *conditioned lifted BP*, the first approach for computing arbitrary joint marginals using lifted BP. It relates conditioning to computing shortest-paths. Exploiting this link in order to establish runtime bounds is an interesting avenue for future work. By combining lifted BP and variable conditioning, it can readily be applied to models of realistic domain size. As our results show significant efficiency gains are obtainable, sometimes order of magnitude, compared to naively running (lifted) BP in each iteration. An interesting avenue for future work is to apply CLBP within important AI tasks such as finding the MAP assignment, sequential forward sampling, and structure learning. Furthermore, our results suggest to develop lifted cutset conditioning algorithms, see e.g. (Bidyuk and Dechter, 2007), and to lift Eaton and Ghahrmani's (2009) fast heuristic for selecting nodes to be clamped to improve CLBP's accuracy.

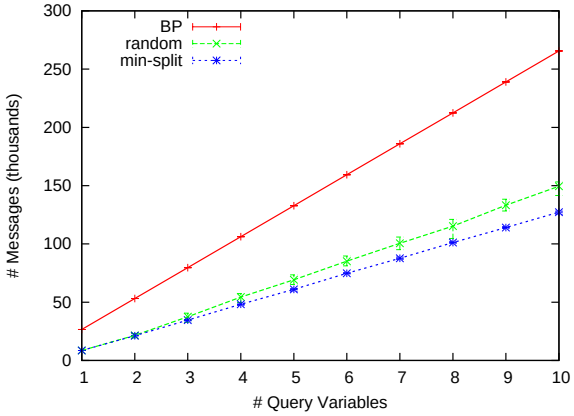


Figure 4: Number messages sent for computing joint marginals of varying size for BP, "random" and "min-split" order CLBP.

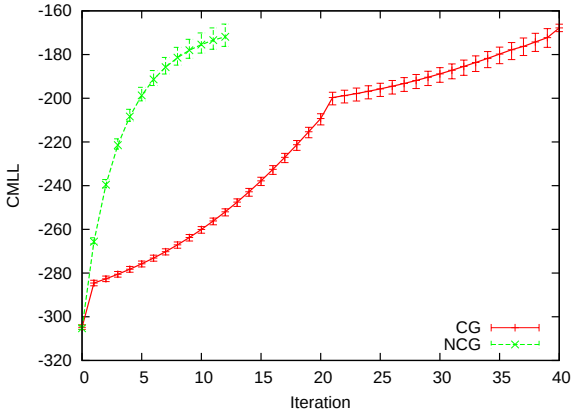


Figure 5: Learning curves for "Friends-and-Smokers" MLN. Optimization using clause covariances shows faster convergence.

Acknowledgements. This work was supported by the Fraunhofer ATTRACT fellowship STREAM and by the European Commission under contract number FP7-248258-First-MM.

References

U. Acar, A. Ihler, R. Mettu, and O. Sumer. 2008. Adaptive inference on general graphical models. In *Proc. of the Twenty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, Corvallis, Oregon. AUAI Press.

B. Bidyuk and R. Dechter. 2007. Cutset sampling for bayesian networks. *JAIR*, 28.

A. L. Delcher, A. J. Grove, S. Kasif, and J. Pearl. 1996. Logarithmic-time updates and queries in probabilistic networks. *JAIR*, 4:37–59.

F. Eaton and Z. Ghahramani. 2009. Choosing a variable to clamp: Approximate inference using conditioned belief propagation. In *Proc. of the 12th International Conference on Artificial Intelligence and Statistics (AIStats-09)*.

A.T. Ihler, J.W. Fisher III, and A.S. Willsky. 2005. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936.

K. Kersting, B. Ahmadi, and S. Natarajan. 2009. Counting belief propagation. In J. Bilmes A. Ng, editor, *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI–09)*, Montreal, Canada, June 18–21.

B. Milch, L. Zettlemoyer, K. Kersting, M. Haimes, and L. Pack Kaelbling. 2008. Lifted Probabilistic Inference with Counting Formulas. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI-08)*, July 13-17.

J. Mooij, B. Wemmenhove, H. Kappen, and T. Rizzo. 2007. Loop corrected belief propagation. In *Proc. of the 11th International Conference on Artificial Intelligence and Statistics (AIStats-09)*.

Joris M. Mooij. 2009. libDAI 0.2.3: A free/open source C++ library for Discrete Approximate Inference. http://www.libdai.org/.

A. Nath and P. Domingos. 2010. Efficient lifting for online probabilistic inference. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*.

J. Pearl. 1991. *Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2. edition.

M. Richardson and P. Domingos. 2006. Markov Logic Networks. *MLJ*, 62:107–136.

P. Sen, A. Deshpande, and L. Getoor. 2009. Bisimulation-based approximate lifted inference. In J. Bilmes A. Ng, editor, *Proc. of the 25th Conference on Uncertainty in Artificial Intelligence (UAI–09)*, Montreal, Canada, June 18–21.

P. Singla and P. Domingos. 2008. Lifted First-Order Belief Propagation. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI-08)*, pages 1094–1099, July 13-17.

M. Welling and Y.W. Teh. 2003. Linear response for approximate inference. In *Proc. of NIPS-03*, pages 191–199.

Sub-quadratic Markov tree mixture learning based on randomizations of the Chow-Liu algorithm

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@univ-nantes.fr, philippe.leray@univ-nantes.fr

François Schnitzler and Louis Wehenkel
Department of EECS & GIGA-Research,
Grande Traverse, 10 - B-4000 Liège - Belgium
fschnitzler@ulg.ac.be, L.Wehenkel@ulg.ac.be

Abstract

The present work analyzes different randomized methods to learn Markov tree mixtures for density estimation in very high-dimensional discrete spaces (very large number n of discrete variables) when the sample size (N) is very small compared to n. Several sub-quadratic relaxations of the Chow-Liu algorithm are proposed, weakening its search procedure. We first study naïve randomizations and then gradually increase the deterministic behavior of the algorithms by trying to focus on the most interesting edges, either by retaining the best edges between models, or by inferring promising relationships between variables. We compare these methods to totally random tree generation and randomization based on bootstrap-resampling (bagging), of respectively linear and quadratic complexity. Our results show that randomization becomes increasingly more interesting for smaller N/n ratios, and that methods based on simultaneously discovering and exploiting the problem structure are promising in this context.

1 Introduction

Directed probabilistic graphical models encode a joint distribution over a set of variables by a product of conditional probability distributions, one for each variable conditionally to its parents in the directed graph. These models may be learned from data and used to perform probabilistic inferences over the encoded distribution (Pearl, 1986). However, exact inference and learning with such models are both NP-hard, unless the skeleton of the graph is constrained (Cooper, 1990). Existing learning algorithms are not scalable to high dimensional spaces because of their excessive computational complexity (Auvray and Wehenkel, 2002).

Markov Trees are an interesting subclass of directed graphical models, whose skeletons are acyclic and for which each node of the graph has (at most) one parent. With Markov trees, the computational complexity of probabilistic inference and parameter learning are linear in the number of variables (Pearl, 1986). Further, Markov tree structures may be learned efficiently by the Chow-Liu algorithm (Section 3.1), quadratic in the number of variables.

While Markov trees impose strong modeling restrictions, mixtures of Markov trees can represent a much wider (actually unlimited) class of probability densities than single Markov trees while retaining their interesting computational properties in terms of inference (Meila and Jordan, 2000), making these models attractive for scaling graphical models to high-dimensional problems. As a matter of fact, these simple graphical models were used for these reasons,

Pp. 17–25 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

in order to build optimized mixtures of models for probability density estimation (Meila and Jordan, 2000) by using an expectation-minimization algorithm.

In 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 dataset and on the other hand to combine in some appropriate fashion an ensemble of predictors obtained by multiple iterations of the so perturbed search algorithm. This approach may in particular lead to a strong reduction in variance (Breiman, 1996). It was first explored for probability density estimation in (Ammar et al., 2008) by comparing various kinds of large ensembles of simple graphical models in the form of Markov Trees.

The above mentioned algorithms for learning mixtures of Markov trees use the Chow-Liu algorithm (Chow and Liu, 1968). However, since this algorithm is quadratic in the number of variables, these methods do not scale well to very high-dimensional problems, with thousands or even millions of variables. Thus, (Schnitzler et al., 2010; Ammar et al., 2010) tried to investigate mixtures of models learned by using various randomized versions of the Chow-Liu algorithm, with the aim of reducing the computational complexity below the quadratic level, and simultaneously improving accuracy in small (i.e. realistic) sample size conditions by variance reduction. The aim of the present paper is to analyse these methods and compare their results in the same framework.

The rest of the paper is organized as follows. We describe tree models and models of mixtures of trees more formally in section 2, and state in section 3 the different tree mixture learning algorithms that we want to analyse. We explain and discuss our empirical evaluation of these algorithms in section 4, before concluding.

2 Mixtures of Markov trees

Let $X = \{X_1, \ldots, X_n\}$ be a finite set of discrete random variables, and $D = (x^1, \cdots, x^N)$ be a sample (we will use the term "dataset" to denote it) of joint observations $x^i = \{x_1^i, \cdots, x_n^i\}$ independently drawn from some data-generating density $\mathbb{P}_G(X_1, \ldots, X_n)$.

A mixture distribution $\mathbb{P}_{\hat{\mathcal{T}}}(X_1, \ldots, X_n)$ induced by a multiset $\hat{\mathcal{T}} = \{T_1, \ldots, T_m\}$ of m Markov trees is defined as a convex combination of elementary Markov tree densities, i.e.

$$\mathbb{P}_{\hat{\mathcal{T}}}(X) = \sum_{i=1}^{m} \mu_i \mathbb{P}_{T_i}(X),$$

where $\mu_i \in [0, 1]$, $\sum_{i=1}^{m} \mu_i = 1$, and $\mathbb{P}_{T_i}(X)$ is the probability density over X encoded by the graphical model composed of the Markov tree structure S_i and its parameter set $\tilde{\theta}_i$:

$$\mathbb{P}_{T_i}(X) = \mathbb{P}_{S_i, \tilde{\theta}_i}(X) = \prod_{p=1}^{n} P_{\tilde{\theta}_i}(X_p | Pa_{S_i}(X_p)),$$

where $Pa_{S_i}(X_p)$ is the parent variable of X_p in the tree structure S_i.

Several versions of Markov tree mixtures were studied in (Ammar et al., 2009; Ammar et al., 2008) as an alternative to classical methods of density estimation in the context of high-dimensional spaces and small datasets: mixtures of tree structures generated in a totally randomized fashion with linear complexity in the number of variables and ensembles of optimal trees derived from bootstrap replicas of the dataset by the Chow and Liu algorithm (Chow and Liu, 1968) (i.e. bagging of Markov trees).

Other studies tried to relax the Chow-Liu algorithm to reduce its computational complexity while maintaining its accuracy (Schnitzler et al., 2010; Ammar et al., 2010). In the present work we analyze these methods in terms of computational complexity, accuracy, and running time, within a same framework.

3 Panel of learning algorithms

Algorithm 1 describes our general methodology to learn a mixture of m Markov trees from a dataset D.

It may be declined by using different variants of the three subroutines it uses, namely *BuildMarkovTreeStructure*, *LearnPars*, and *Comp-*

Weights. In this paper we focus on the effect of varying only the first one, used to infer the structures S_i of the mixture terms. Next, we describe the different versions of *BuildMarkovTreeStructure* that we have considered in our study. We start by describing the original Chow-Liu method.

Algorithm 1 (Learning a Markov tree mixture)**.**

1. *Repeat for* $i = 1, \cdots, m$*:*
   (a) $S_i = BuildMarkovTreeStructure(D)$
   (b) $\tilde{\theta}_i = LearnPars(S_i, D)$
2. $(\mu)_{i=1}^m = CompWeights((S_i, \tilde{\theta}_i)_{i=1}^m, D)$
3. *Return* $\left(\mu_i, S_i, \tilde{\theta}_i\right)_{i=1}^m$

3.1 Chow-Liu algorithm

This algorithm learns a Markov tree structure maximizing the likelihood of the training set (Chow and Liu, 1968). Its principle is described by Algorithm 2. It can be decomposed in two steps : Step 1. computes from the dataset the maximum likelihood estimates of the mutual informations between each pair of variables to fill an $n \times n$ symmetrical matrix (MI); Step 2. searches for a maximum weight spanning tree (MWST) in this matrix (e.g. Kruskal's algorithm (Cormen et al., 2001), used here).

Algorithm 2 (Chow-Liu algorithm)**.**

1. $MI = [0]_{n \times n}$*; Repeat for* $k = 1, \cdots, n$*:*
   Repeat for $j = k+1, \cdots, n$*:*
   i. $MI[k, j] = CompMI(X_k, X_j, D)$*;*
   ii. $MI[j, k] = MI[k, j]$.
2. $S = CompKruskal(MI)$*; Return* S.

The first step requires $\mathcal{O}(n^2 N)$ computations, while the second has a complexity of $E \log(E)$ with E the number of considered edges. In the Chow-Liu algorithm $E = n(n-1)/2$, so this second complexity becomes $\mathcal{O}(n^2 \log(n^2))$.

3.2 Randomized edge sampling

To reduce the complexity of the Chow-Liu algorithm, we propose to apply the Perturb and Combine principle by learning each model from an incomplete matrix MI.

The random edge sampling algorithm performs this by randomly selecting a subset of a priori fixed size K of different pairs of variables according to a uniform distribution. These terms are used to partially fill the matrix MI used as input to the MWST algorithm. Algorithm 3 describes this procedure.

Algorithm 3 (randomized edge sampling)**.**

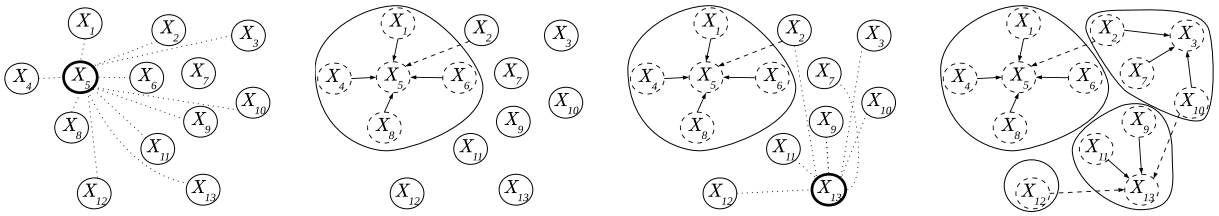
1. $MI = [0]_{n \times n}$ *; Repeat for* $k = 1, \cdots, K$*:*
   (a) *Draw new random pair* $(i_1, i_2) \in \{1, \ldots, n\}^2$*;*
   (b) $MI[i_1, i_2] = CompMI(X_{i_1}, X_{i_2}, D)$*;*
   (c) $MI[i_2, i_1] = MI[i_1, i_2]$.
2. $S = CompKruskal(MI)$*; Return* S.

The complexity of Algorithm 3 is loglinear in the number K of edges drawn. Notice that the tree structure that it infers may be disconnected, and that its dependence on the dataset is increasing with the value of K. We will report in this paper simulations and results for two values of the parameter : $K = n\ log(n)$ considered in (Ammar et al., 2010), and $K = 0.33n(n-1)/2$. The first value allows a total complexity of $n \log(n) \log(n \log(n))$, which is sub-quadratic and very close to the quasi-linear. The second corresponds approximately to the edges sampled by the method described in the next section.

3.3 Randomized vertex clustering

Another idea to weaken the Chow and Liu procedure was proposed by (Schnitzler et al., 2010). This method is a less naïve approach to sampling the matrix MI, which targets potentially interesting (i.e. of large weight) edges. Algorithm 4 details this two-step process, that first builds a local structure of the problem, and then focuses on pairs of variables located close to each other in that structure.

The first step consists in an approximate online clustering of the variables based on their mutual information, inspired by leader clustering (a cluster C_p is represented by its leader L_p). As illustrated in figure 1, a sequence C of clusters is created until all variables belong to one. The construction of a cluster C_p is based on two thresholds on mutual information: one cluster-threshold (MI_C) and one neighborhood-threshold (MI_N). The cluster is built by comparing the mutual information of each remaining unclustered variable to the new leader (L_p,



(a) First a leader (here X_5) is chosen at random and compared to all 12 other variables.

(b) Next, the 1st cluster is built. Here it is made of 5 members and has one neighbor.

(c) The 2nd leader (X_{13}, the farthest from X_5) is compared only to 7 variables.

(d) Final result, after 4 iterations. All edges considered are kept for the MWST inference.

Figure 1: Illustration of the vertex clustering algorithm.

chosen first at random, then among unclustered variables by minimizing $\sum_{q<p} MI(L_p, L_q)$). An unclustered variable X is identified as:

1. member of C_p, if $MI(X, L_p) > MI_C$,
2. neighbor of C_p, if $MI_C \geqslant MI(X, L_p) > MI_N$,
3. not related to C_p, otherwise.

Setting those thresholds can be seen as excluding potentially independent variables. This exclusion rate can be controlled, since the maximum likelihood estimate of the mutual information for two independent variables asymptotically follows a χ^2 law. In this work, its percentile 0.5 (respectively 5) was used for MI_C (MI_N).

Algorithm 4 (Vertex Clustering)**.**

1. $\mathcal{V} = \mathcal{X}$; $C = \emptyset$; $MI = [0]_{n\times n}$; *Repeat until* $\mathcal{V} = \emptyset$:
   (a) $L = GetNewLeader(\mathcal{V}, C)$;
   (b) $C, MI \mathrel{+}= MakeCluster(L, \mathcal{V}, MI_N, MI_C)$.
2. $nbClusters = size(C)$;
   Repeat for $p = 1, \cdots, nbClusters$:
   (a) *Repeat for* $i_1, i_2 : X_{i_1}, X_{i_2} \in C_p$:
      i. $MI[i_1, i_2] = CompMI(X_{i_1}, X_{i_2}, D)$;
      ii. $MI[i_2, i_1] = MI[i_1, i_2]$.
   (b) *Repeat* $\forall q < p : C_q \in Neibhors(C_p)$:
      Repeat for $i_1, i_2 : X_{i_1} \in C_p, X_{i_2} \in C_q$:
      A. $MI[i_1, i_2] = CompMI(X_{i_1}, X_{i_2}, D)$;
      B. $MI[i_2, i_1] = MI[i_1, i_2]$.
3. $S = CompKruskal(MI)$; *Return* S.

In the second step of the algorithm, the mutual information of all potentially interesting pairs (X_i, X_j) are computed and used as edge-weights for the MWST algorithm. Interesting pairs (a) are in the same cluster or (b) span two neighboring clusters, i.e one variable of one cluster is a neighbor of the other cluster. In addition, all edges evaluated during the clustering process are used as candidate edges.

The complexity of this algorithm is between linear and quadratic in the number of variables, depending on the numerical values of MI_C and MI_N and the problem structure.

3.4 Inertial search heuristic

This algorithm (Ammar et al., 2010) for computing a sequence of sub-optimal MWST was designed to improve the base method from section 3.2. In this work we also apply it to the vertex clustering algorithm (section 3.3).

The inertial method takes advantage of the Markov tree structure S_{i-1} built in the previous iteration to partially fill the new MI_i matrix. The weights (recomputed in case bootstrap copies of the dataset are used) of the edges of the Markov tree built at the previous iteration $i-1$ are first written in the MI_i matrix of the current iteration i, and a new set of edges generated by the base method (either at random or by vertex clustering) is inserted afterwards. This is described by Algorithm 5.

The complexity of this method is similar to the base method.

Algorithm 5 (Inertial research procedure)**.**

1. $MI_i = [0]_{n\times n}$;
   Repeat for $k = 1, \cdots, nbEdges(S_{i-1})$:
   (a) $(i_1, i_2) = GetIndices(GetEdge(S_{i-1}, k))$;
   (b) $MI_i[i_1, i_2] = CompMI(X_{i_1}, X_{i_2}, D)$;
   (c) $MI_i[i_2, i_1] = MI_i[i_1, i_2]$.
2. *Repeat for* $k = 1, \cdots, nbEdges(BaseMethod)$:
   (a) $(i_1, i_2) =$ *indices of edge* k *from BaseMethod;*
   (b) $MI_i[i_1, i_2] = CompMI(X_{i_1}, X_{i_2}, D)$;
   (c) $MI_i[i_2, i_1] = MI_i[i_1, i_2]$.
3. $S_i = CompKruskal(MI_i)$; *Return* S_i.

3.5 Other variants

Two other variants (baselines) were also considered, namely random trees and bagging.

The first one draws a tree structure totally at random (i.e. independently from the dataset) through the use of Prüfer lists. Complexity is linear in n (Ammar et al., 2008).

Bagging can also be used to increase randomization in a given method, by supplying a bootstrap replica of the original dataset to any tree-structure learning algorithm. This may actually be quite productive in order to randomize tree structures (see the results below). However, as far as accuracy is concerned, it turns out to be preferable to use the full dataset for parameter estimation of each Markov tree generated by this method (Schnitzler et al., 2010).

4 Empirical simulations

We apply the algorithm variants described in Section 3 to synthetic problems to assess their performance. We carried out repetitive experiments for different data-generating (or target) densities as described in Section 4.1; our results are reported in Section 4.2.

4.1 Experimental protocol

We present here results obtained on 10 different target distributions over $n = 1000$ binary variables, and we report them for mixtures and datasets of various sizes.

Target density generation Target densities are synthetic distribution factorizing according to a general directed acyclic graph structure. These models (structure and parameters) are generated by the algorithm described in (Ammar et al., 2008).

Datasets We focus our analysis on rather small datasets ($N = 100, 250, 1000$) with respect to the number $n = 1000$ of variables. This replicates the usual situation in high-dimensional problems, which are the motivation of this work. For each considered sample size and for each target distribution, we generate 5 different datasets.

Mixture learning For a given dataset, and for a given tree structure learning algorithm, we apply the mixture learning algorithm (Algo 1) by generating ensemble models of growing sizes ($m = 1, m = 10, \ldots, m = 150$) in order to appraise the effect of the ensemble size on the quality of the resulting model.

In all our simulations, the parameters of the Markov tree models are learned from the dataset by maximum a posteriori estimation using uniform Dirichlet priors.

In our empirical tests we have always weighted the individual terms uniformly (i.e. $\mu = 1/m$ in Algorithm 1).

Accuracy evaluation We assess the quality of each generated mixture by the Kullback-Leibler divergence (Kullback and Leibler, 1951), an asymmetric measure of similarity of a given distribution $\mathbb{P}_{\hat{\mathcal{T}}}$ to a target distribution $\mathbb{P}_G$, defined by

$$D_{KL}(\mathbb{P}_G \,||\, \mathbb{P}_{\hat{\mathcal{T}}}) = \sum_{X \in \mathcal{X}} \mathbb{P}_G(X) \log_2 \left(\frac{\mathbb{P}_G(X)}{\mathbb{P}_{\hat{\mathcal{T}}}(X)} \right).$$

Since screening all 2^{1000} configurations of $\mathcal{X}$ is not possible, we estimate this quantity by Monte Carlo using a random sample of configurations generated according to $\mathbb{P}_G$:

$$\hat{D}_{KL}(\mathbb{P}_G \,||\, \mathbb{P}_{\hat{\mathcal{T}}}) = \sum_{X \sim \mathbb{P}_G} \log_2 \left(\frac{\mathbb{P}_G(X)}{\mathbb{P}_{\hat{\mathcal{T}}}(X)} \right).$$

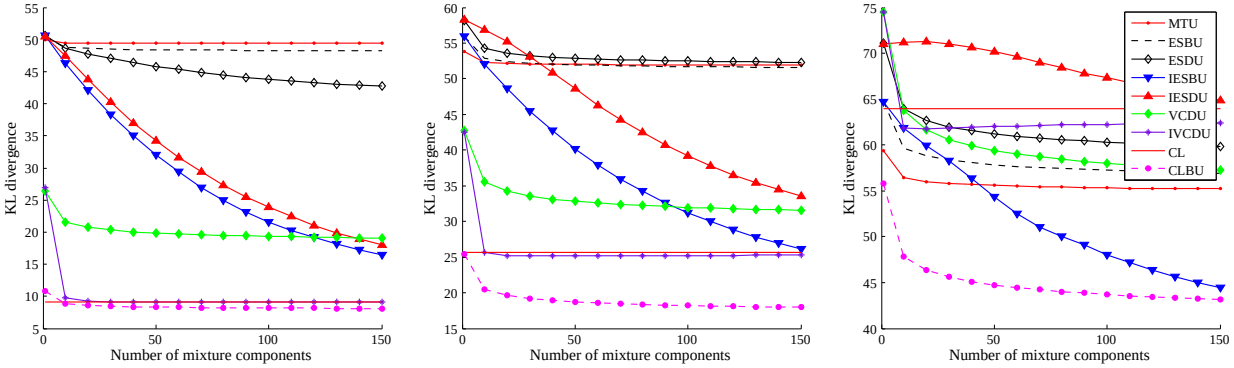
In this work, we generated for each data-generating distribution and each learning algorithm a fixed set of 50000 samples, which is then used for the Monte Carlo estimation of D_{KL} of the models produced by the algorithm applied to the datasets issued from this distribution and with a growing number of mixture terms m.

4.2 Results and discussion

Table 1 describes the algorithm variants that we have evaluated, recalls their computational complexities, and also gives indications of their relative computing times in our implementation. The performances of these algorithms in terms of accuracy (D_{KL} estimates) are reported in Figures 2 and 3. As reference method we use the Chow and Liu single tree method (denoted by CL in the table and figures).

Table 1: In the names of the algorithms we study, D means no alteration to the dataset and B the use of bootstrap replica. m, n, K stand for the number of terms in the mixture, of variables and of sampled pairs of variables. U emphasizes the fact that we are using uniform weights in the mixture. (CPU times are given for $n = 1000$ variables)

Name	Tree generation	Dataset	Complexity	running time (one tree)
MTU	random	D	mn	0.0017
ESBU	rand. Edge Samp.	B	$mK\log(K)$	0.02
ESDU	rand. Edge Samp.	D	$mK\log(K)$	0.02
IESBU	Inertial Edge Samp.	B	$mK\log(K)$	0.02
IESDU	Inertial Edge Samp.	D	$mK\log(K)$	0.02
IESDU%	Inertial Edge Samp.	D	$(K = 165000)$	0.72
VCDU	Vert. Clust	D	up to $mn^2\log(n)$	0.92
IVCDU	Inertial Vert. Clust	D	up to $mn^2\log(n)$	0.92
CL	Chow-Liu	D	$n^2\log(n)$	1
CLBU	Chow-Liu	B	$mn^2\log(n)$	1



(a) 1000 samples. (b) 250 samples. (c) 100 samples.

Figure 2: Average performance of the algorithms described in Table 1 on 5 target distributions of 1000 variables times 10 datasets, with sample sizes decreasing.

Figures 2(a) and 3 display the resulting $\hat{D}_{KL}$ values for growing mixture sizes m on datasets of 1000 samples. From Fig. 2(a) we observe that vertex clustering (VCDU) seems far better than random edge sampling (ESDU).

The comparison is however not fair, because VCDU samples approximately 33% of all edges at each iteration, and ESDU only 1.4%. To provide a more accurate comparison, the latter was modified (ESDU%, in Figure 3 and Table 1) to use the same number of edges than VCDU. The results exposed in Figure 3 with this setting confirm that VCDU is also superior to ESDU%.

The inertial heuristic can be used to enhance both methods (IESDU, IVCDU) with nearly no additional complexity cost (see Table 1), leading to an increase in the improvement rate with the size of the model. Figures 2(a) and 2(b) show that IVCDU converges to a model slightly better than the CL tree (the best edges have been found, so the optimal tree is always learned), while the inertial randomized methods (IRB and IRS) with a complexity close to quasi-linear tend to approach CL when the number of mixture components grows and surpasses IVCDU which complexity is higher.

The same convergence can be observed for IESDU% (IESDU on 33% of edges instead of 1.4%) in Fig. 3, which also converges to the CL tree, albeit slower than IVCDU.

ESDU is degraded by the use of bagging (ESBU), and both methods yield performance only slightly better than random structures (MTU). We therefore conjecture that the low quality (as opposed to the low number) of edges used in ESDU introduces too much randomiza-

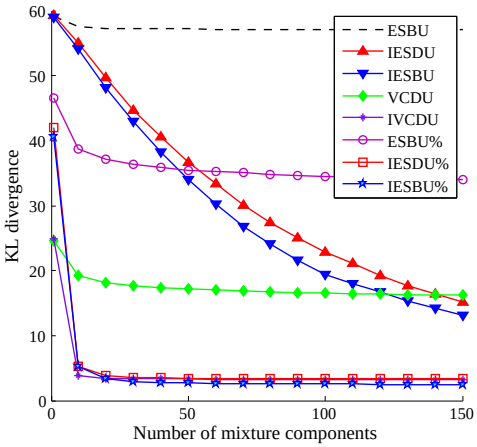


Figure 3: A comparison between vertex clustering and random edge sampling methods, where the latter are modified to use the same number of edges than the first, shows that vertex clustering is still clearly superior. (1000 samples)

tion to benefit from bagging. Indeed, when the quality of the edges considered increases over time (IESDU) bagging actually ameliorates the method. This hints towards that considering the use of bagging methods with CL trees or with the inertial vertex clustering could probably be improved by computing the best edges (i.e. significant on the original dataset) only once instead of repeating it for every new tree.

Experiments performed on sample sets of size 250 and 100 are reported in Fig. 2(b) and 2(c). Decreasing the number of samples from 1000 to 250 does not modify the results very much, as illustrated in Fig. 2(b). The main difference lies in the relative performance of all methods compared to CL algorithm, which seem to improve faster as m increases. This tends to indicate that randomization is increasingly more beneficial as the number of samples decreases.

The observation of the behavior of the bagging methods in Figure 2 further confirms this analysis. The gap between pairs of methods using the same algorithm on the original dataset or on bootstrap replicas (CL - CLBU, ESDU - ESBU, IESDU - IESBU) is widening when the number of samples is decreasing. Bagging of Chow-Liu trees (CLBU) is actually the best method on all dataset sizes. However, using bagging with inertial procedures and edge sampling yields also impressive results: the curve IESBU converges to CLBU in Fig. 2(c), while the IESBU algorithm has a much lower complexity than CLBU.

An alternative way to understand these results is in terms of over-fitting. From Fig. 2(c) we can see that IVCDU first surpasses CL after a few iterations, but the addition of subsequent terms worsens the mixture, which converges to the CL curve. Actually, at that point the model has fully learned the optimal tree, and it is repeatedly added to the model. The rise of the curve at that point signals the over-fitting. In addition, we can notice that MTU, the method using structures drawn independently from the dataset behaves better than most other methods in this context.

Observing the first tree of each model in the same figure, we can observe over-fitting again. IESBU (and ESBU) is better than IESDU (and ESDU). Likewise, the first term of CLBU is better than CL. A tree structure learned on a perturbed dataset leads to a graphical model that is more general.

This behavior is still present at 250 samples, but is no longer noticeable at 1000 : IESBU and IESDU start at the same point, and the first bagged tree is worse than the optimal.

The application of BDeu weights (not reported in this paper) to the methods presented here leads to similar conclusions. But we found out that many methods actually display worse performances in terms of accuracy when combined with such Bayesian weights. This is understandable, since weighting each structure by its posterior probability makes sense for MTU only, since asymptotically only in this context the mixture will converge to a canonical Bayesian method. The methods that benefit the most from those 'Bayesian' weighting scheme are IRSBU and IRSDU. The first terms (built with few information) are gradually eliminated in favor of those identified later on (where the inertial procedure has iteratively improved the quality of the edges).

5 Conclusions and future works

In this paper, we have compared several randomization methods aiming to approximate the Chow-Liu algorithm, with the objective of reducing its computational complexity in the context of learning mixtures of trees, and motivated by the variance reduction potential of randomization in the context of learning in high-dimensional problems.

Based on our results on synthetic experiments, we claim that, in real conditions, i.e. when the number of samples is much smaller than the number of variables, randomization is interesting for probability density estimation in the form of mixtures of Markov Trees. That interest actually increases when the number of samples goes down, or when the dimensionality of the space is increasing.

In addition, we have shown that exploiting the structure of the problem by focussing on strong edges leads to methods able to compete in terms of performance with more time-consuming procedures like bagging.

We therefore plan to keep investigating this approach. In particular, a candidate area for improvement is the transmission of knowledge between terms. Increasing the number of reused edges might speed up the convergence. Another direction of research would be the consideration of continuous variables, and the consideration of a priori known dependency/independency structures for the given problem.

Acknowledgments

This work was supported by FRIA/FNRS Belgium, Wallonie Bruxelles International, the French ministry of foreign and European affairs, the MESR in the framework of Hubert Curien partnerships, the BioMaGNet IUAP network of the Belgian Science Policy Office and the Pascal2 NOE of the EC-FP7. The scientific responsibility rests with the authors.

References

S. Ammar, Ph. Leray, B. Defourny, and L. Wehenkel. 2008. High-dimensional probability density estimation with randomized ensembles of tree structured Bayesian networks. In *Proceedings of the fourth European Workshop on Probabilistic Graphical Models (PGM'08)*, pages 9–16, Hirtshals, Denmark.

S. Ammar, Ph. Leray, B. Defourny, and L. Wehenkel. 2009. Probability density estimation by perturbing and combining tree structured Markov networks. In *Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2009)*, pages 156–167, Verona, Italy.

S. Ammar, Ph. Leray, and L. Wehenkel. 2010. Subquadratic Markov tree mixture models for probability density estimation. In *19th International Conference on Computational Statistics (COMPSTAT 2010)*, pages 673–680, Paris, France.

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.

L. Breiman. 1996. Arcing classifiers. Technical report, Dept. of Statistics, University of California.

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

G.F. Cooper. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, March.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2001. *Introduction to Algorithms, Second Edition.* MIT Press and McGraw-Hill.

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

M. Meila and M. I. Jordan. 2000. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48.

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

F. Schnitzler, Ph. Leray, and L. Wehenkel. 2010. Towards sub-quadratic learning of probability density models in the form of mixtures of trees. In *18th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2010)*, pages 219–224, Bruges, Belgium.

Learning CB-decomposable Multi-dimensional Bayesian Network Classifiers

Hanen Borchani, Concha Bielza and Pedro Larrañaga
Departamento de Inteligencia Artificial, Facultad de Informática
Universidad Politécnica de Madrid, Boadilla del Monte, 28660, Madrid, Spain.
hanen.borchani@upm.es, {mcbielza, pedro.larranaga}@fi.upm.es

Abstract

Multi-dimensional Bayesian network classifiers (MBCs) have been recently introduced to deal with multi-dimensional classification problems where instances are assigned to multiple classes. MBCs have a restricted topology partitioning the set of class and feature variables into three different subgraphs: class subgraph, feature subgraph and bridge subgraph. In this paper, we propose a novel learning algorithm for class-bridge (CB) decomposable MBCs into maximal connected components. Basically, based on a wrapper greedy forward selection approach, the algorithm firstly learns the bridge and feature subgraphs. Then, while the number of components is greater than one and there is an accuracy improvement, it iteratively and sequentially merges together the components, and updates the bridge and feature subgraphs. By learning CB-decomposable MBCs, the computations of MPE are alleviated comparing to general MBCs. Experimental comparison with state-of-the-art algorithms are carried out using synthetic and real-world data sets. The obtained results show the merits of our proposed algorithm.

1 Introduction

Multi-dimensional classification (van der Gaag and de Waal, 2006) is an extension of the classical one-dimensional classification where each instance given by a vector of m features $\mathbf{x} = (x_1, ..., x_m)$ is associated, with not only a single class value, but with a set of d class values $\mathbf{c} = (c_1, ..., c_d)$. Multi-dimensional classification has been motivated by several application domains. For instance, in text categorization, a text document may be assigned to more than one topic; in scene classification, each semantic scene may be assigned to several classes, such as beach, sunset and mountain; in medical diagnosis, a patient may be suffering from multiple diseases, etc.

In recent years, the concept of multi-dimensionality has been introduced in Bayesian network classifiers (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007; Rodríguez and Lozano, 2008; Bielza et al., 2010). In these probabilistic graphical models, known as multi-dimensional Bayesian network classifiers (MBCs), the graphical structure partitions the set of class and feature variables into three different subgraphs: class subgraph, feature subgraph and bridge subgraph, and the parameter set defines the conditional probability distribution of each variable given its parents.

One of the most challenging problems with MBC models involves the most probable explanation (MPE) computation, which is known to be NP-hard in general, and presents a significant complexity especially when the MBC has a large number of class variables.

In this paper, in order to alleviate the MPE computational burden, we consider the family of class-bridge decomposable multi-dimensional Bayesian network classifiers (CB-decomposable MBCs) introduced by Bielza et al. (2010). In fact, by decomposing class and bridge subgraphs of an MBC graphical structure into r maximal connected components, the maximization problem for MPE computation can

Pp. 25–33 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

be transformed into r maximization problems operating in lower dimensional spaces. Moreover, using CB-decomposable MBCs may provide more insight about the domain and better interpretability of learned structures than large and complex MBCs which have no explicit representation for domain decomposability.

However, CB-decomposable MBCs merits have been only discussed and proved theoretically in (Bielza et al., 2010), and no learning approach nor an experimental study have been presented to empirically demonstrate the usefulness of this new family of MBCs.

In order to tackle these shortcomings, we propose in the present work a novel algorithm for learning CB-decomposable MBCs based on a wrapper greedy forward selection approach. Broadly speaking, in a first phase our algorithm learns a CB-decomposable MBC with a number of maximal connected components equal to the number of class variables. This is carried out by learning a selective naive Bayes (Langley and Sage, 1994) for each class variable C, then, removing their possible common children to have an initial bridge subgraph and the corresponding CB-decomposable MBC. In a second phase, a feature subgraph defining dependence relationships between the set of feature variables is learned. Finally, in a third phase, while the number of maximal connected components is greater than one and there is an accuracy improvement, the algorithm iteratively and sequentially merges together the components, then updates the bridge and feature subgraphs.

The remainder of this paper is organized as follows. In Section 2 we review the definitions of MBCs and CB-decomposable MBCs. In Section 3 we describe our algorithm for learning CB-decomposable MBCs from data. In Section 4 we present experimental set up and results. Finally, we round off the paper with some conclusions in Section 5.

2 Multi-dimensional Bayesian Network Classifiers

A Bayesian network over a set of discrete random variables $\mathbf{U} = \{X_1, ..., X_n\}, n \geq 1$, is a pair $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$. $\mathcal{G} = (V, A)$ is a directed acyclic graph (DAG) whose vertices V correspond to variables $\mathbf{U}$, and whose arcs A represent direct dependencies between the vertices. $\mathbf{\Theta}$ is a set of conditional probability distributions such that $\theta_{x_i|\mathbf{pa}(x_i)} = p(x_i \mid \mathbf{pa}(x_i))$ defines the conditional probability of each possible value x_i of X_i given a set value $\mathbf{pa}(x_i)$ of $\mathbf{Pa}(X_i)$, where $\mathbf{Pa}(X_i)$ denotes the set of parents of X_i in $\mathcal{G}$.

A Bayesian network $\mathcal{B}$ represents a joint probability distribution over $\mathbf{U}$ factorized according to structure $\mathcal{G}$ as follows:

$$p(X_1, ..., X_n) = \prod_{i=1}^{n} p(X_i \mid \mathbf{Pa}(X_i)) \cdot \quad (1)$$

Definition 1. A *multi-dimensional Bayesian network classifier* (MBC) is a Bayesian network $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$ where the structure $\mathcal{G} = (V, A)$ has a restricted topology. The set of vertices V is partitioned into two sets: $V_C = \{C_1, ..., C_d\}, d \geq 1$, of class variables and $V_X = \{X_1, ..., X_m\}, m \geq 1$, of feature variables $(d + m = n)$. Moreover, the set of arcs A is partitioned into three sets A_C, A_X and A_{CX}, such that:

- $A_C \subseteq V_C \times V_C$ is composed of the arcs between the class variables having a subgraph $\mathcal{G}_\mathcal{C} = (V_C, A_C)$ -*class subgraph*- of $\mathcal{G}$ induced by V_C.
- $A_X \subseteq V_X \times V_X$ is composed of the arcs between the feature variables having a subgraph $\mathcal{G}_\mathcal{X} = (V_X, A_X)$ -*feature subgraph*- of $\mathcal{G}$ induced by V_X.
- $A_{CX} \subseteq V_C \times V_X$ is composed of the arcs from the class variables to the feature variables having a subgraph $\mathcal{G}_{\mathcal{CX}} = (V, A_{CX})$ -*bridge subgraph*- of $\mathcal{G}$ connecting class and feature variables.

Classification with an MBC under a 0-1 loss function amounts to solving the most probable explanation (MPE) problem, i.e. for a given evidence $\mathbf{x} = (x_1, ..., x_m)$ we have to get:

$$\begin{aligned} \mathbf{c}^* &= (c_1^*, ..., c_d^*) \\ &= arg \max_{c_1,...,c_d} p(C_1 = c_1, ..., C_d = c_d \mid \mathbf{x}) \cdot \end{aligned} \quad (2)$$

Example 1. Figure 1 shows an example of an MBC structure $\mathcal{G} = \mathcal{G}_{\mathcal{C}} \cup \mathcal{G}_{\mathcal{X}} \cup \mathcal{G}_{\mathcal{CX}}$ where the set of class variables $V_C = \{C_1, C_2, C_3, C_4\}$ and the set of feature variables $V_X = \{X_1, X_2, X_3, X_4, X_5, X_6\}$. We have:

$$\begin{aligned}&\max_{c_1,...,c_4} p(C_1 = c_1, ..., C_4 = c_4 \mid \mathbf{x}) \\ &\propto \max_{c_1,...,c_4} p(c_1)p(c_2 \mid c_1)p(c_3)p(c_4) \\ &\cdot p(x_1 \mid c_1, x_2)p(x_2 \mid c_1, c_2)p(x_3 \mid c_3) \\ &\cdot p(x_4 \mid c_3)p(x_5 \mid c_4, x_1, x_6)p(x_6 \mid c_3, x_3)\end{aligned}$$

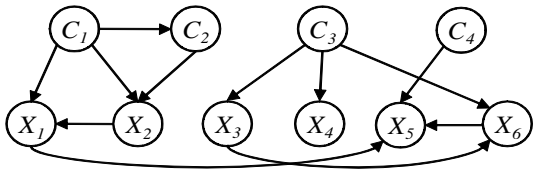


Figure 1: An example of an MBC structure.

Note that depending on the graphical structures of the class and feature subgraphs we can differentiate between several families of MBCs. Such families can be denoted as `class subgraph structure-feature subgraph structure` MBC where the possible structures of each subgraph may be: empty, tree, polytree, or DAG. MBC families used in the literature include `tree-tree` MBC (van der Gaag and de Waal, 2006), `polytree-polytree` MBC (de Waal and van der Gaag, 2007), `DAG-empty` MBC (Qazi et al., 2007), and `DAG-DAG` MBC (Rodríguez and Lozano, 2008). In this paper, we do not consider any restrictions on the learned MBC structures, i.e. any possible structure type is allowed for either class or feature subgraphs.

Definition 2. A *class-bridge decomposable multi-dimensional Bayesian network classifier* (CB-decomposable MBC) is an MBC $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$ where the class subgraph $\mathcal{G}_{\mathcal{C}}$ and bridge subgraph $\mathcal{G}_{\mathcal{CX}}$ are decomposed into r maximal connected components, such that

1. $\mathcal{G}_C \cup \mathcal{G}_{(CX)} = \bigcup_{i=1}^{r} (\mathcal{G}_{C_i} \cup \mathcal{G}_{(CX)_i})$, where $\mathcal{G}_{C_i} \cup \mathcal{G}_{(CX)_i}$, with $i = 1, ..., r$, are its r *maximal connected components*, and
2. $Ch(V_{C_i}) \cap Ch(V_{C_j}) = \emptyset$, with $i, j = 1, ..., r$ and $i \neq j$, where $Ch(V_{C_i})$ denotes the children of all the variables in V_{C_i}, the subset of class variables in $\mathcal{G}_{C_i}$ (*non-shared children property*).

Bielza et al. (2010) proved that the MPE computation can be alleviated thanks to MBC class-bridge decomposability. In fact, maximizing over the set of all class variables amounts to maximizing over each class variable subset of the identified maximal connected components, i.e. maximizing over lower dimensional subspaces than originally.

Theorem 1. *Given a CB-decomposable MBC where $\mathcal{I}_i$ represents the sample space associated with V_{C_i}, then*

$$\begin{aligned}&\max_{c_1,...,c_d} p(C_1 = c_1, ..., C_d = c_d \mid \boldsymbol{x}) \\ &\propto \prod_{i=1}^{r} \max_{\boldsymbol{c}^{\downarrow V_{C_i}} \in \mathcal{I}_i} \Bigg(\prod_{C \in V_{C_i}} p(c \mid \boldsymbol{pa}(c)) \\ &\quad \cdot \prod_{X \in Ch(V_{C_i})} p(x \mid \boldsymbol{pa}_{V_C}(x), \boldsymbol{pa}_{V_X}(x)) \Bigg), \qquad (3)\end{aligned}$$

where $\boldsymbol{c}^{\downarrow V_{C_i}}$ represents the projection of vector $\boldsymbol{c}$ to the coordinates found in V_{C_i}. $\boldsymbol{Pa}_{V_C}(X)$ and $\boldsymbol{Pa}_{V_X}(X)$ denote, respectively, the class parents and feature parents of X in $\mathcal{G}$. Obviously, for any class variable C, we have $\boldsymbol{Pa}_{V_X}(C) = \emptyset$ and $\boldsymbol{Pa}_{V_C}(C) = \boldsymbol{Pa}(C)$.

Given $\mathbf{x}$, each expression to be maximized in Equation (3) will be denoted as $\phi_i^{\mathbf{x}}(\mathbf{c}^{\downarrow V_{C_i}})$, $i = 1, ..., r$, i.e.

$$\begin{aligned}\phi_i^{\mathbf{x}}(\mathbf{c}^{\downarrow V_{C_i}}) &= \prod_{C \in V_{C_i}} p(c \mid \mathbf{pa}(c)) \\ &\cdot \prod_{X \in Ch(V_{C_i})} p(x \mid \mathbf{pa}_{V_C}(x), \mathbf{pa}_{V_X}(x)) \cdot \qquad (4)\end{aligned}$$

It holds that $\phi_i^{\mathbf{x}}(\mathbf{c}^{\downarrow V_{C_i}}) \propto p(\mathbf{C}^{\downarrow V_{C_i}} = \mathbf{c}^{\downarrow V_{C_i}} \mid \mathbf{x})$.

Example 2. Let us reconsider the MBC shown in Figure 1. It is a CB-decomposable MBC with $r = 3$. Its three maximal connected components are depicted in Figure 2. The first one is $\mathcal{G}_{C_1} \cup \mathcal{G}_{(CX)_1}$ with $V_{C_1} = \{C_1, C_2\}$ and

$Ch(V_{C_1}) = \{X_1, X_2\}$, the second is $\mathcal{G}_{C_2} \cup \mathcal{G}_{(CX)_2}$ with $V_{C_2} = \{C_3\}$ and $Ch(V_{C_2}) = \{X_3, X_4, X_6\}$, and the third is $\mathcal{G}_{C_3} \cup \mathcal{G}_{(CX)_3}$ with $V_{C_3} = \{C_4\}$ and $Ch(V_{C_3}) = \{X_5\}$. Note that $Ch(V_{C_1}) \cap Ch(V_{C_2}) = Ch(V_{C_1}) \cap Ch(V_{C_3}) = Ch(V_{C_2}) \cap Ch(V_{C_3}) = \emptyset$, as required. As a maximization problem we get:

$$\begin{aligned}
&\max_{c_1,...,c_4} \; p(C_1 = c_1, ..., C_4 = c_4 \mid \mathbf{x}) \\
&= \max_{c_1,c_2} p(c_1)p(c_2 \mid c_1)p(x_1 \mid c_1, x_2)p(x_2 \mid c_1, c_2) \\
&\quad \cdot \max_{c_3} p(c_3)p(x_3 \mid c_3)p(x_4 \mid c_3)p(x_6 \mid c_3, x_3) \\
&\quad \cdot \max_{c_4} p(c_4)p(x_5 \mid c_4, x_1, x_6) \\
&= \max_{c_1,c_2} \phi_1^{\mathbf{X}}(c_1, c_2) \cdot \max_{c_3} \phi_2^{\mathbf{X}}(c_3) \cdot \max_{c_4} \phi_3^{\mathbf{X}}(c_4) \cdot
\end{aligned}$$

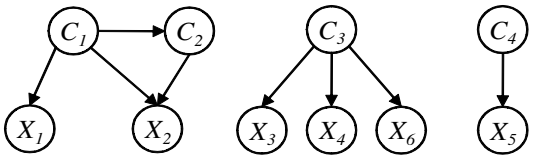


Figure 2: The three maximal connected components of the MBC example of Figure 1.

3 Learning CB-decomposable MBCs from Data

We describe in this section our proposed algorithm for learning CB-decomposable MBCs from data based on a wrapper greedy forward selection approach. Let $\mathcal{D}$ be a data set of N observations containing a value assignment for each variable $X_1, ..., X_m, C_1, ..., C_d$, i.e. $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{c}^{(1)}), ..., (\mathbf{x}^{(N)}, \mathbf{c}^{(N)})\}$. As performance metrics, we use:

1. The *mean accuracy* over the d class variables:

$$Acc_m = \frac{1}{d} \sum_{i=1}^{d} \frac{1}{N} \sum_{l=1}^{N} \delta(c'_{li}, c_{li}), \qquad (5)$$

where $\delta(c'_{li}, c_{li}) = 1$ if $c'_{li} = c_{li}$, and 0 otherwise. Note that c'_{li} denotes the C_i class value outputted by the MBC for case l and c_{li} is its corresponding true value.

2. The *global accuracy* over the d-dimensional class variable:

$$Acc_g = \frac{1}{N} \sum_{l=1}^{N} \delta(\mathbf{c}'_l, \mathbf{c}_l), \qquad (6)$$

where $\delta(\mathbf{c}'_l, \mathbf{c}_l) = 1$ if $\mathbf{c}'_l = \mathbf{c}_l$, and 0 otherwise. That is, we call for a complete equality between all the components of the vector of predicted classes and the vector of real classes.

Our learning algorithm consists of three main phases, outlined by Algorithm 1, and detailed in what follows. Note that, in the different phases, the classifier accuracy is denoted Acc, which is equal to Acc_m or Acc_g depending on using the mean or the global accuracy.

3.1 Phase I: Learn bridge subgraph

Starting from an empty graphical structure, the first step in this phase is learning a selective naive Bayes (Langley and Sage, 1994) for each class variable C_i, $i = 1, ..., d$.

The d resulting selective naive Bayes models represent $r = d$ maximal connected components that may have common children. Thus, the next step is to check the non-shared children property in order to induce an initial CB-decomposable MBC. This is accomplished by removing, if necessary, all common children, based on two criteria, namely, the feature insertion rank and the accuracy.

Let $rank_j^i$ denotes the insertion rank of feature X_j in the selective naive Bayes NB_i for C_i, and $rank_j^k$ denotes the insertion rank of feature X_j in the selective naive Bayes NB_k for C_k. $rank_j^i < rank_j^k$ means that X_j is firstly selected by NB_i. Hence, in this case, X_j will be kept in NB_i and removed from NB_k. Otherwise, and in case that $rank_j^i = rank_j^k$, we proceed to compare the accuracies Acc^i and Acc^k, denoting respectively the accuracy of NB_i and NB_k when X_j was included in, then keep X_j in the NB presenting the highest accuracy and remove it from the other.

The result of this phase is a simple CB-decomposable MBC, denoted as $CB\text{-}MBC_r^b$, where only the bridge subgraph is defined and the class and feature subgraphs are still empty.

Algorithm 1

Input: $\mathcal{D}$, T
Output: $CB\text{-}MBC_r^{bfc}$
$\mathcal{G}_C = \emptyset$; $\mathcal{G}_{CX} = \emptyset$; $\mathcal{G}_X = \emptyset$; $r = d$.

[**Phase I: Learn bridge subgraph**]
Learn selective naive Bayes NB_i, $i = 1, ..., r$.
for Each NB_i, NB_k having a common feature X_j **do**
  if $rank_j^i < rank_j^k$ **then**
    Remove X_j from NB_k.
  else
    if $rank_j^i = rank_j^k$ **then**
      if $Acc^i > Acc^k$ **then**
        Remove X_j from NB_k.
      else
        Remove X_j from NB_i.
      end if
    end if
    Remove X_j from NB_i.
  end if
end for
Obtain $\mathcal{G}_{CX} = \bigcup_{i=1}^{r}(NB_i)$, that is, $CB\text{-}MBC_r^b$.

[**Phase II: Learn feature subgraph**]
for $TrialNumber = 1 : T$ **do**
  Add randomly one arc to $\mathcal{G}_X$.
  if No accuracy improvement **then**
    Discard the arc and do not consider it in subsequent iterations.
  end if
end for
Obtain $CB\text{-}MBC_r^{bf}$.

[**Phase III: Merge maximal connected components**]
$CB\text{-}MBC_r^{bfc} \longleftarrow CB\text{-}MBC_r^{bf}$.
Stop = False.
while $r > 1$ and *not Stop* **do**
  for Each class variables C_i, C_k pertaining to two different maximal connected components **do**
    Evaluate the arc insertion from C_i to C_k.
  end for
  Select the arc with the best accuracy Acc^{r-1}.
  if $Acc^{r-1} > Acc^r$ **then**
    Update $\mathcal{G}_C$.
    $Acc^r = Acc^{r-1}$.
    $CB\text{-}MBC_r^{bfc} \longleftarrow CB\text{-}MBC_{r-1}^{bfc}$.
    $r = r - 1$.
    while Accuracy improvement **do**
      Update $\mathcal{G}_{CX}$: add an arc from a class to a feature of the new merged component.
    end while
    while Accuracy improvement **do**
      Update $\mathcal{G}_X$: add an arc between feature variables.
    end while
  else
    Stop = True.
  end if
end while
return $CB\text{-}MBC_r^{bfc}$.

3.2 Phase II: Learn feature subgraph

This phase consists of learning the feature subgraph by introducing the dependence relationships between the feature variables. Since it may be impractical to consider all possible arc additions between the feature variables, especially if the number of features m is large, we will fix a parameter T as a maximum number of iterations.

In each iteration, an arc is selected at random between a pair of feature variables. If there is an accuracy improvement, the arc is added to $\mathcal{G}_X$, otherwise it is discarded and will not be considered in subsequent iterations. This phase ends when T is reached, and the induced MBC is denoted as $CB\text{-}MBC_r^{bf}$.

Note that, thanks to MBC decomposability, the classification accuracy associated with the arc addition in each iteration can be evaluated in a straightforward and local way. In fact, after adding an arc from X_i to X_j, only the term corresponding to variable X_j changes, that is, only the MPE computation of the maximal connected component to which X_j pertains, changes and needs to be reevaluated. The MPE computation over all remaining maximal connected components remains unchanged, which considerably reduces the computational burden.

3.3 Phase III: Merge maximal connected components

Taking as input the CB-decomposable MBC found in the previous phase, having r maximal connected components and a corresponding accuracy denoted Acc^r, the third phase consists of learning the class subgraph, which leads to merging the maximal connected components of the current CB-decomposable MBC, then updating the bridge and feature subgraphs.

As a first step, all possible arc additions between the class variables pertaining to different maximal connected components are evaluated. If there is an accuracy improvement, i.e. $Acc^{r-1} > Acc^r$, the subgraph $\mathcal{G}_C$ is updated by adding the arc improving the accuracy the most, and r is reduced to $r - 1$ maximal connected components.

Subsequently, a bridge update step is performed inside the new induced maximal connected component. Dependence relationships that may be added from class to feature variables of the corresponding component are greedily evaluated, and only when there is an accuracy improvement, the best one is added.

Note that, once again, the MBC decomposability plays a key role in alleviating the complexity of MPE computation since each possible arc addition between class variables or from class to feature variables is evaluated locally. Moreover, for bridge updating step, the MPE is only recomputed for the new merged component, which alleviates more the complexity.

The last step in this phase consists of updating the feature subgraph by inserting, one by one, additional arcs between feature variables while this improves the accuracy.

This phase iterates over these three steps, and terminates when no more component merging can improve the accuracy or until the condition $r = 1$ is reached. A CB-decomposable MBC denoted as $CB\text{-}MBC_r^{bfc}$ is returned.

4 Related Work

In this section, we briefly review the state-of-the-art on MBCs learning algorithms.

Van der Gaag and de Waal (2006) decompose the learning problem of `tree-tree` MBCs into two separate optimization problems: first learning the class subgraph using Chow and Liu's algorithm (1968), then, given a fixed bridge subgraph, learning the feature subgraph using also Chow and Liu's algorithm. The bridge subgraph is selected using a wrapper approach garanteeing a high classifier accuracy.

De Waal and van der Gaag (2007) present a theoretical approach for learning `polytree-polytree` MBCs where class and feature subgraphs are separately learnt based on Rebane and Pearl's algorithm (1989). Nevertheless, the induction of the bridge subgraph was not specified.

Moreover, Qazi et al. (2007) learn `DAG-empty` MBCs where the class subgraph is induced by standard Bayesian networks procedures, the bridge subgraph is learnt by adding dependence relationships form each class variable to a subset of selected features, and the feature subgraph is kept empty.

Rodríguez and Lozano (2008) use a multi-objective evolutionary approach to learn `DAG-DAG` MBCs. Each permitted MBC structure is coded as an individual with three substrings, one per subgraph. Based on different classification rules, joint and marginal, they define the objective functions as k-fold cross-validated estimators of each class classification error. The aim is to find non-dominated structures according to the objective functions.

More recently, Bielza et al. (2010) propose different learning algorithms, namely, pure filter (guided by the K2 algorithm), pure wrapper (guided by the classification accuracy) and hybrid algorithm (a combination of pure filter and pure wrapper), allowing any Bayesian network structure in the three MBC subgraphs.

Similarly as Bielza et al. (2010), we have no constraints about the subgraph structures of the generated MBCs. However, contrary to their leaning algorithms and contrary to other existing works, our proposal is to learn the new family of CB-decomposable MBCs instead of learning general MBCs.

5 Experiments

In order to evaluate our learning algorithm, we firstly perform experiments with a synthetic data set. We randomly generate an MBC, containing 6 class and 10 feature binary variables, decomposed into 3 maximal connected components. Then, we randomly sample a data set of size 1000 using the probabilistic logic sampling method (Henrion, 1988). We apply our algorithm denoted as `CB-MBC`, and other four algorithms, namely, `Tree-Tree` (van der Gaag and de Waal, 2006), `Polytree-Polytree` (de Waal and van der Gaag, 2007), `Pure Filter` (Bielza et al., 2010) and `Pure Wrapper` (Bielza et al., 2010), all starting from an empty structure.

We consider both the mean and the global accuracy to learn and then evaluate the performance of the classifiers. Furthermore, in order

to test the ability of the classifiers to recover the initial MBC structure, we compare each learned structure (LS) to the initial one (IS) using the following structural evaluation metrics:

- M1: percentage of arcs in LS that are present in IS, i.e. percentage of correctly-found arcs.
- M2: percentage of arcs in LS that are absent in IS, i.e. percentage of superfluous arcs.
- M3: percentage of arcs in IS that are oriented in an opposite direction in LS, i.e. percentage of badly-oriented arcs.
- M4: percentage of arcs in IS that are absent in LS, i.e. percentage of missing arcs.

Five-fold cross-validation experiments are run for each learning algorithm. Table 1 shows the average results over these runs.

Table 1: Experimental results over the synthetic data set.

Mean accuracy					
Classifier	Acc_m	M1	M2	M3	M4
`CB-MBC`	0.7182	26.66	37.55	11.88	61.44
`Tree-Tree`	0.7129	30.55	49.44	5.55	63.89
`Polytree-Polytree`	0.6330	30.10	15.10	6.21	63.66
`Pure Filter`	0.5351	7.55	14.66	8.88	83.55
`Pure Wrapper`	0.7098	22.22	42.88	17.77	60.00
Global accuracy					
Classifier	Acc_g	M1	M2	M3	M4
`CB-MBC`	0.2877	11.55	14.66	1.33	87.10
`Tree-Tree`	0.2838	25.00	53.88	8.33	66.66
`Polytree-Polytree`	0.2845	28.99	15.10	5.44	65.55
`Pure Filter`	0.2160	7.99	15.55	7.54	84.44
`Pure Wrapper`	0.2800	16.00	48.44	14.22	69.77

Our algorithm outperforms the state-of-the-art algorithms in terms of mean and global accuracy. For structural evaluation, `Tree-Tree` and `Polytree-Polytree` present the best percentages of correctly-found arcs (M1) while `Pure Filter` has the lowest one. `Tree-Tree` and `Pure Wrapper` induce the highest percentages of superfluous arcs (M2), and `Pure Wrapper` also induces a high percentage of badly-oriented arcs (M3) comparing to the rest of the algorithms. Moreover, we may observe that, with global accuracy, the learned structures are sparser, leading to more important percentages of missing arcs (M4) for all learning algorithms.

As additional experiments, we consider the real data set Emotions (Trohidis et al., 2008). It is about a multi-dimensional classification of music into emotions. It contains 72 music features for 593 songs categorized into one or more out of 6 classes of emotions: amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely, and angry-aggressive.

As previously, the accuracies of the considered learning algorithms are computed using 5-fold cross-validation. The results are summarized in Table 2. Note that, with this real data set, we do not have an initial MBC structure, so the structural evaluation is omitted in this set of experiments.

Table 2: Experimental results over Emotions data set.

Classifier	Acc_m	Acc_g
`CB-MBC`	0.8326	0.3639
`Tree-Tree`	0.8135	0.2977
`Polytree-Polytree`	0.8052	0.3422
`Pure Filter`	0.6733	0.2690
`Pure Wrapper`	0.8293	0.3650

From Table 2, we may conclude that our algorithm performs well. In fact, with the mean accuracy, `CB-MBC` presents the best accuracy, while with the global accuracy, `Pure Wrapper` slightly outperforms `CB-MBC`.

Finally, in Figure 3, we plot the computation learning time of the various learning algorithms, using the mean and global accuracy, for both synthetic and Emotions data sets.

Clearly, the algorithms using a filter approach require less computation than those using a wrapper approach. Moreover, the computation time of our algorithm is lower than the other wrapper approaches, mainly over the synthetic data set, which is basically due to the MBC CB-decomposability and the alleviation of MPE computation. Note also that the computation time with global accuracy is lower, since it is more difficult to improve the learned models, so that the algorithm ends in earlier iterations.

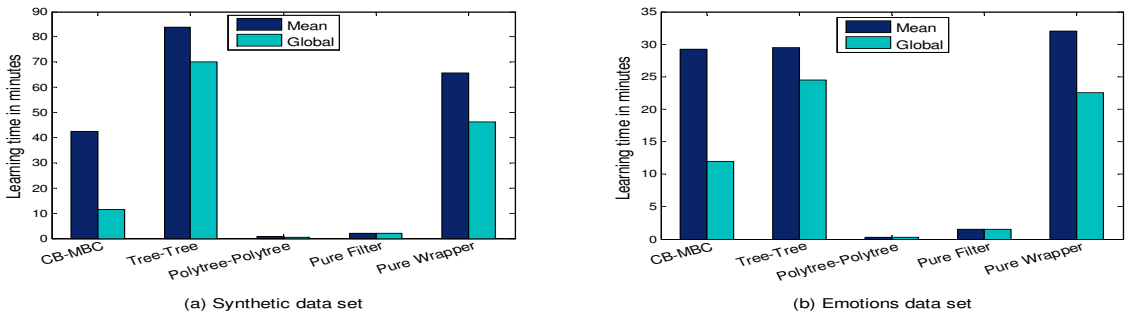


Figure 3: Computation learning times over (a) synthetic data set and (b) Emotions data set.

6 Conclusion

In this paper, we proposed a novel algorithm for learning CB-decomposable MBCs from data based on a wrapper forward selection approach. Indeed, CB-decomposability allows the alleviation of MPE computations. Experimental results with both synthetic and real-world data sets show that our algorithm performs well and requires less computation time than the state-of-the-art wrapper learning algorithms.

In the future, we intend to carry out additional experiments and investigate possible improvements of our algorithm. For instance, we intend to test the alternation between forward and backward selection techniques, and study the use of a filter approach mainly for feature subgraph learning in order to avoid the random arc additions between features. Furthermore, it would be interesting to extend our algorithm to deal with incremental learning from new incoming data, i.e. updating the current CB-decomposable MBC over time without a need to relearn it from scratch. In case of non-stationary domains, this may also require a detection mechanism to monitor the concept drift.

Acknowledgements

Work supported by projects TIN2007-62626 and Cajal Blue Brain (Spanish Ministry of Science and Innovation) and by project Dynamo (FONCICYT, European Union and Mexico).

References

C. Bielza, G. Li and P. Larrañaga. 2010. Multi-dimensional classification with Bayesian networks. Technical Report UPM-FI/DIA/2010-1, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid.

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

P.R. de Waal and L.C. van der Gaag. 2007. Inference and learning in multi-dimensional Bayesian network classifiers. *In Proceedings of the Ninth European Conference on Symbolic and Quantitative Approaches to Reasoning under Uncertainty, Lecture Notes in Artificial Intelligence, Springer*, 4724:501–511.

M. Henrion. 1988. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. *In Proceedings of the Fourth Conference on the Uncertainty in Artificial Intelligence*, pages 149–163.

P. Langley and S. Sage. 1994. Induction of selective Bayesian classifiers. *In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406.

M. Qazi, G. Fung, S. Krishnan, R. Rosales, H. Steck, R.B. Rao, D. Poldermans and D. Chandrasekaran. 2007. Automated heart wall motion abnormality detection from ultrasound images using Bayesian networks. *In International Joint Conference on Artificial Intelligence*, pages 519–525.

G. Rebane and J. Pearl. 1989. The recovery of causal polytrees from statistical data. *In Proceedings of the Third Conference on Uncertainty in Artificial Intelligence*, pages 222–228.

J.D. Rodríguez and J.A. Lozano. 2008. Multi-objective learning of multi-dimensional Bayesian classifiers. *In Proceedings of the Eighth International Conference on Hybrid Intelligent Systems*, pages 501–506.

K. Trohidis, G. Tsoumakas, G. Kalliris and I. Vlahavas. 2008. Multilabel classification of music into emotions. *In Proceedings of the International Society for Music Information Retrieval Conference*, pages 325–330.

L.C. van der Gaag and P.R. de Waal. 2006. Multi-dimensional Bayesian network classifiers. *In Proceedings of the Third European Conference on Probabilistic Graphical Models*, pages 107–114.

Bayesian Networks on Dirichlet Distributed Vectors

Wray Buntine and Lan Du
NICTA and Australian National University
{wray.buntine, lan.du}@nicta.com.au

Petteri Nurmi
University of Helsinki
ptnurmi@cs.helsinki.fi

Abstract

Exact Bayesian network inference exists for Gaussian and multinomial distributions. For other kinds of distributions, approximations or restrictions on the kind of inference done are needed. In this paper we present generalized networks of Dirichlet distributions, and show how, using the two-parameter Poisson-Dirichlet distribution and Gibbs sampling, one can do approximate inference over them. This involves integrating out the probability vectors but leaving auxiliary discrete count vectors in their place. We illustrate the technique by extending standard topic models to "structured" documents, where the document structure is given by a Bayesian network of Dirichlets.

1 Introduction

The Gaussian and the multinomial distributions are the only ones allowing exact Bayesian inference over arbitrary Bayesian networks. Other distributions can be included as long as they stay at particular nodes and restrictions are placed on the direction of inference. For instance, mixing multinomial and Gaussian works as long as in all cases of inference the multinomials are strictly non-descendents of the Gaussian variables (Lauritzen, 1989). Extending inference to Monte Carlo or Gibbs sampling, and allowing general purpose samplers dramatically broadens the range of distributions one can allow (Thomas et al., 1992).

In this paper we show how to perform approximate inference over networks of probability vectors related via (approximate) Dirichlet distributions. So for instance, one could have a hidden Markov model where the hidden states are probabilities vectors $\boldsymbol{p}_i$ related in a chain by:

$$\boldsymbol{p}_{i-1} \sim \text{Dirichlet}\left(\beta \boldsymbol{p}_i\right), \ \boldsymbol{p}_i \sim \text{Dirichlet}\left(\beta \boldsymbol{p}_{i+1}\right).$$

Nested Dirichlet distributions like this would make an ideal tool for Bayesian modelling of hierarchical clustering, for instance, where previous methods have hierarchically partitioned the features (Hanson et al., 1991). They can also have a significant role in various places in extending standard topic models (Buntine and Jakulin, 2006; Blei et al., 2003), for instance, making documents, topics or components hierarchical. The standard model is in Figure 1. Both the links $\boldsymbol{\alpha} \to \boldsymbol{\mu}$ and $\boldsymbol{\gamma} \to \boldsymbol{\phi}$ are Dirichlet but $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are really unknown so should be modelled by Dirichlets themselves, yielding a nested Dirichlet.

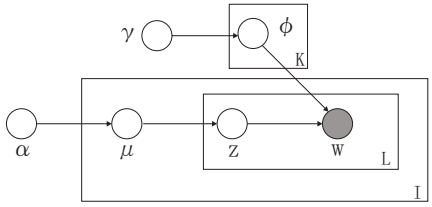


Figure 1: Standard Latent Dirichlet Allocation (LDA) Model. $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are Dirichlet priors, $\boldsymbol{\mu}$ is a distribution over topics, $\boldsymbol{\phi}$ is a distribution over words, and z is a topic associated with a word w. I indicates the number of documents, and L denotes the number of words in document i. K is the number of topics.

While MAP solutions for nested Dirichlets do exist, and an MCMC sampler on the real-valued probability vectors could also be applied, the

Pp. 33–41 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

nested Dirichlets are often used as latent variables, so these solutions are inaccurate and inefficient respectively. This paper develops a more efficient and exact Gibbs sampler exists that integrates out the real-valued probability vectors by introducing small-valued integer vectors instead.

Bayesian hierarchical methods often use the *two-parameter Poisson-Dirichlet process* (PDP), also known as the Pitman-Yor process (Ishwaran and James, 2001). In Section 2, we discuss these models from our perspective and how they can be used in nested Dirichlet modelling. The basic theory comes from (Buntine and Hutter, 2010), some borrowed from (Teh, 2006a). Using these tools, in Section 3 networks of probability vectors distributed as Dirichlet and using PDPs, first presented in (Wood and Teh, 2009), are shown along with techniques for their statistical analysis. Their analysis is the main contribution of this paper. Some examples of these networks are embedded in new versions of topic models. These are presented, and some empirical results given in Section 4.

2 The Two-parameter Poisson-Dirichlet Process

The major tool used here is the *two-parameter Poisson-Dirichlet process* (PDP), also known as the Pitman-Yor process, which is an extension of the *Dirichlet process* (DP). The PDP is defined as: $\boldsymbol{\nu} \sim PDP(\boldsymbol{\mu}, a, b)$, where a is a discount parameter, b is a strength parameter, and $\boldsymbol{\mu}$ is a base distribution for $\boldsymbol{\nu}$. These are used as tools for non-parametric and hierarchical Bayesian modelling.

The general theory of PDPs applies them to arbitrary measurable spaces (Ishwaran and James, 2001), for instance real valued spaces, but in many recent applications, such as language and vision applications, the domain is countable (*e.g.*, "English words") and standard theory requires some modifications. In language domains, PDPs and DPs are proving useful for full probability modelling of various phenomena including n-gram modelling and smoothing (Teh, 2006b; Goldwater et al., 2006; Mochihashi and Sumita, 2008), dependency models for grammar (Johnson et al., 2007; Wallach et al., 2008), and for data compression (Wood et al., 2009). The PDP-based n-gram models correspond well to versions of Kneser-Ney smoothing (Teh, 2006b), the state of the art method in applications. These models are intriguing from the probability perspective, as well as sometimes being competitive with performance based approaches. More generally, PDPs have been applied to clustering (Rasmussen, 2000) and image segmentation (Sudderth and Jordan, 2009).

For our purposes, knowledge of the details of the PDP and DP are not required. The two key results needed, however, follow. The Dirichlet Approximation Lemma is adapted from (Buntine and Hutter, 2010):

Lemma 1 (Dirichlet Approximation Lemma)**.** *Given a K-dimensional probability vector $\boldsymbol{\mu}$, the following approximations on distributions hold (as $a \to 0$)*

$$\begin{aligned} PDP(0, b, discrete_K(\boldsymbol{\mu})) &\approx Dirichlet_K(b\boldsymbol{\mu}) \ , \\ PDP(a, 0, discrete_K(\boldsymbol{\mu})) &\approx Dirichlet_K(a\boldsymbol{\mu}) \ . \end{aligned}$$

The first approximation is justified because the means and the first two central moments (orders 2 and 3) of the LHS and RHS distributions are equal. The second approximation is justified because the mean and first two central moments (orders 2 and 3) agree with error $O(a^2)$.

Using this, we can see that one can replace the K-dimensional distribution $\text{Dirichlet}_K(b\boldsymbol{\mu})$ by its approximation $\text{PDP}(0, b, \text{discrete}_K(\boldsymbol{\mu}))$, and this greatly simplifies reasoning because PDPs turn out to be conjugate to multinomials. This result follows from the Marginalisation Lemma adapted from (Buntine and Hutter, 2010) and originally proven in a different format in a hierarchical context by Teh (Teh, 2006a).

Lemma 2 (Marginalisation Lemma)**.** *Given a probability vector $\boldsymbol{\mu}$ of dimension K, and the following set of priors and likelihoods for $j = 1, ..., J$*

$$\begin{aligned} \boldsymbol{\nu}_j &\sim PDP(a, b, discrete_K(\boldsymbol{\mu})) \\ \boldsymbol{n}_j &\sim multinomial_K(\boldsymbol{\nu}_j, N_j) \end{aligned}$$

where $N_j = \sum_k n_{j,k}$. *Introduce auxiliary latent variables* $\boldsymbol{t}_j$ *such that* $t_{j,k} \leq n_{j,k}$ *and* $t_{j,k} = 0$ *if and only if* $n_{j,k} = 0$, *then the following posterior distribution holds which marginalises out the* $\boldsymbol{\nu}_{1:J}$ *but introduces auxillary variables* $\boldsymbol{t}_{1:J}$*:*

$$p(\boldsymbol{n}_{1:J}, \boldsymbol{t}_{1:J} | a, b, \boldsymbol{\mu}) \;=\; \prod_j C_{\boldsymbol{n}_j}^{N_j} \frac{(b|a)_{\sum_k t_{j,k}}}{(b)_{N_j}} \prod_{j,k} S_{t_{j,k},a}^{n_{j,k}} \prod_k \mu_k^{\sum_j t_{j,k}} \;. \tag{1}$$

The functions introduced in the lemma are as follows: $C_{\boldsymbol{n}_j}^{N_j}$ is the multi-dimensional choose function of a multinomial; $(x)_N$ is given by $(x|1)_N$, $(x|y)_N$ denotes the Pochhammer symbol with increment y, it is defined as

$$\begin{aligned}(x|y)_N &= x(x+y)...(x+(N-1)y) \\ &= \begin{cases} x^N & \text{if } y = 0, \\ y^N \times \frac{\Gamma(x/y+N)}{\Gamma(x/y)} & \text{if } y > 0, \end{cases}\end{aligned}$$

where $\Gamma(\cdot)$ denotes the standard gamma function; and $S_{M,a}^N$ is a generalised Stirling number given by the linear recursion (Buntine and Hutter, 2010; Teh, 2006a)

$$S_{M,a}^{N+1} = \; S_{M-1,a}^N + (N - Ma) S_{M,a}^N$$

for $M \leq N$. It is 0 otherwise and $S_{0,a}^N = \delta_{N,0}$. These rapidly become very large so computation needs to be done in log space using a logarithmic addition.

In summary, we replace the Dirichlet by an approximation, a PDP, and then since thus is conjugate to a multinomial (with introduction of suitable auxiliary variables), it allows ready processing in nested contexts. The nesting is proven in the next section. We can apply Lemma 2 without having any clear interpretation of the auxiliary variables $t_{j,k}$, which would require a more extensive presentation of PDPs[1].

3 Networks of Dirichlet Distributed Vectors

Assume a directed graph G composed of nodes indexed by integers $1, ..., J$ with probability vectors $\boldsymbol{\nu}_j$ for $j = 1, ..., J$ and directed edges (i, j) for $i \in E_j$ where E_j is the set of parents of the node j. *Parameters* in the model are as follows:

$\boldsymbol{\nu}_j$: the vector of probabilities at the j-th node.

$\boldsymbol{\rho}_j$: the vector of mixing probabilities at each node, gives how parent probability vectors are mixed. Only used when $|E_j| > 1$.

Hyperparameters in the model are:

a, b: parameters for the PDP.

$\boldsymbol{\alpha}_j$: Dirichlet prior parameters for root node j which has no parents.

$\boldsymbol{\gamma}_j$: Dirichlet prior parameters for mixing probabilities $\boldsymbol{\rho}_j$ when $|E_j| > 1$.

Denote this set of hyperparameters as $\mathcal{H}$.

We consider models of the form:

$$\begin{aligned}\boldsymbol{\nu}_j &\sim \text{Dirichlet}_K(\boldsymbol{\alpha}_j) && \text{for } E_j = \emptyset \\ \boldsymbol{\nu}_j &\sim \text{PDP}\left(a, b, \text{discrete}_K\left(\sum_{i \in E_j} \rho_{i,j} \boldsymbol{\nu}_i\right)\right) && \text{for } E_j \neq \emptyset \;.\end{aligned}$$

Note, when $|E_j| = 1$, then the single hyperparameter $\rho_{i,j}$ for $i \in E_j$ is equal to 1, so the sum degenerates to $\boldsymbol{\nu}_i$. The mixing parameters $\boldsymbol{\rho}_j$ can be modelled as $\boldsymbol{\rho}_j \sim \text{Dirichlet}_{E_j}(\boldsymbol{\gamma}_j)$, only needed when $|E_j| > 1$. Also, data is in the form $\boldsymbol{n}_j \sim \text{multinomial}_K(\boldsymbol{\nu}_j, N_j)$.

These models may be embedded in larger networks, for instance they may be the "topic structure" for a topic model.

3.1 Marginalising Parameters

Applying the Marginalisation Lemma at any leaf node j marginalises out the parameter $\boldsymbol{\nu}_j$ and introduces auxiliary variables $\boldsymbol{t}_j$:

$$C_{\boldsymbol{n}_j}^{N_j} \frac{(b|a)_{\sum_k t_{j,k}}}{(b)_{N_j}} \prod_k S_{t_{j,k},a}^{n_{j,k}} \prod_k \left(\sum_{i \in E_j} \rho_{i,j} \nu_{i,k}\right)^{t_{j,k}} \;.$$

So expand the sum using the multinomial identity. This implies decomposing $t_{j,k}$ into parts coming from each of its parents, and call this

[1] In the Chinese Restaurant Process of the sampling of $\boldsymbol{\nu}_j$, $t_{j,k}$ represents the number of tables that have the same dish k. For non-atomic distributions $t_{j,k}$ would almost surely be 1, so no sampling required.

total now $t^p_{j,k}$, and introduce a complementary sum from their children, called $t^c_{j,k}$

$$t^p_{j,k} \;=\; \sum_{i \in E_j} s_{i,j,k} \;, \quad t^c_{j,k} \;=\; \sum_{l : j \in E_l} s_{j,l,k} \;,$$

and let $\boldsymbol{s}_{j,k} = (s_{i,j,k} : i \in E_j)$. Then integrating out ρ_j, one gets

$$C^{N_j}_{\boldsymbol{n}_j} \frac{(b|a)_{\sum_k t^p_{j,k}}}{(b)_{N_j}} \frac{\prod_{i \in E_j} \Gamma\left(\gamma_{i,j} + \sum_k s_{i,j,k}\right)}{\Gamma\left(\sum_i \gamma_{i,j} + \sum_k t^p_{j,k}\right)} \prod_k \left(S^{n_{j,k}}_{t^p_{j,k},a} C^{t^p_{j,k}}_{\boldsymbol{s}_{j,k}} \prod_{i \in E_j} \nu_{i,k}^{s_{i,j,k}} \right) \;,$$

and to this we must add the constraints for the given j and all k

$$t^p_{j,k} \le n_{j,k} \;, \quad t^p_{j,k} = 0 \text{ if and only if } n_{j,k} = 0 \;.$$

Due to the terms $\nu_{i,k}^{s_{i,j,k}}$ occurring in this, the counts $s_{i,j,k}$ can be added to the data for the node for $\boldsymbol{\nu}_i$, $n_{i,k}$, and the procedure applied recursively. One has therefore proven the nested version of the Marginalisation Lemma,

Lemma 3 (Marginalising a Network of Dirichlets)**.** *Given the network of Dirichlets described in this section, introduce counts $s_{i,j,k} \ge 0$ for $i \in E_j$ and all k when $E_j \neq \emptyset$. These have parent and child totals $t^p_{j,k}$ and $t^c_{j,k}$ as above. These must satisfy constraints at each node j on the k-th value of*

$$t^p_{j,k} \quad \le \quad n_{j,k} + t^c_{j,k} \;, \tag{2}$$

$$t^p_{j,k} = 0 \quad \textit{iff} \quad n_{j,k} + t^c_{j,k} = 0 \;. \tag{3}$$

Then marginalising out all parameters $\boldsymbol{\nu}_j$ and $\boldsymbol{\rho}_j$ yields the posterior ($\mathcal{H}$ is hyperparameters)

$$p(\boldsymbol{n}_{1:J}, \boldsymbol{s}_{1:J,1:K} \,|\, \mathcal{H}) \;= \tag{4}$$

$$\prod_{j : E_j = \emptyset} \frac{\prod_k \Gamma\left(\alpha_{j,k} + n_{j,k} + t^c_{j,k}\right)}{\Gamma\left(\sum_k \alpha_{j,k} + \sum_k n_{j,k} + \sum_k t^c_{j,k}\right)} \prod_{j : E_j \neq \emptyset} \frac{(b|a)_{\sum_k t^p_{j,k}}}{(b)_{N_j + \sum_k t^c_{j,k}}} \frac{\prod_{i \in E_j} \Gamma\left(\gamma_{i,j} + \sum_k s_{i,j,k}\right)}{\Gamma\left(\sum_i \gamma_{i,j} + \sum_k t^p_{j,k}\right)} \prod_{j : E_j \neq \emptyset} C^{N_j}_{\boldsymbol{n}_j} \prod_k \left(S^{n_{j,k} + t^c_{j,k}}_{t^p_{j,k},a} C^{t^p_{j,k}}_{\boldsymbol{s}_{j,k}} \right) \;,$$

The key challenge in working with these models is now handling the auxiliary variables $\boldsymbol{s}_{1:J,1:K}$. When Gibbs sampling is done, for instance, the constraints need to be maintained, and in our experience this turns out to be the major complexity.

3.2 Gibbs Sampling

We consider a single discrete item related to the j-th node, so its distribution is over $\{1, ..., K\}$ and has the probability vector $\boldsymbol{\nu}_j$. The previous theory dealt with multinomials, so $\boldsymbol{n}_j \sim \text{multinomial}_K(\boldsymbol{\nu}_j, N_j)$, however, in practice, we may also consider N_j discrete variables with distribution $\text{discrete}_K(\boldsymbol{\nu}_j)$. Their sufficient statistics are also $\boldsymbol{n}_j$, and the difference in the posterior is that the choose term $C^{N_j}_{\boldsymbol{n}_j}$ is removed.

In Gibbs sampling, suppose we are sampling the probability of this item related the j-th node. Then $n_{j,k}$ will be decreased by 1 for some k, and increased by one for another. Given the form of the posterior in Equation (4), it is easy to consider the change in the posterior when one $n_{j,k}$ is increased, lets denote this as

$$p(\text{increment } n_{j,k} \,|\, \boldsymbol{n}_{1:J}, \boldsymbol{s}_{1:J,1:K}, \mathcal{H}) \;= \frac{p(\text{increment } n_{j,k}, \boldsymbol{n}_{1:J}, \boldsymbol{s}_{1:J,1:K} \,|\, \mathcal{H})}{p(\boldsymbol{n}_{1:J}, \boldsymbol{s}_{1:J,1:K} \,|\, \mathcal{H})}$$

Many of the terms simplify due to ratios of Gamma functions. However, the problem arises that either increasing or decreasing $n_{j,k}$ may violate the constraints for node j when $E_j \neq \emptyset$.

3.2.1 Increasing a count

If $n_{j,k}$ is increased by one, then Constraint (3) may be violated for node j. To fix this, we need to increase one of the $s_{i,j,k}$ for $i \in E_j$. If $|E_j| > 1$ then we have a choice and sampling needs to be done. Once the i is chosen, then $s_{i,j,k}$ is increased by one, and subsequently Constraint (3) may now be violated for node i so the process iterates up the network. When considering this, we need to consider the set of ancestors of j reachable along paths where every element j' has equality for the Constraint (3) on the k-th value.

Suppose one is incrementing $n_{j,k}$ and one's choice is to increment the set of counts $s_{i_2,i_1,k}, ..., s_{i_u,i_{u-1},k}$ where for convenience $i_1 = j$. Along this path, the equality of Constraint (3) must hold for $i_1 = j, i_2, ..., i_{u-1}$. Then the probability of incrementing $n_{j,k}$ using this path to balance equality constraints, denoted

$$p(\text{incr } n_{i_1,k}, s_{i_2,i_1,k}, ..., s_{i_u,i_{u-1},k} \,|\, \boldsymbol{n}_{1:J}, \boldsymbol{s}_{1:J,1:K}, \mathcal{H})$$

is given by

$$\left(\frac{\alpha_{i_u,k} + n_{i_u,k} + t^c_{i_u,k}}{\sum_k \alpha_{i_u,k} + \sum_k n_{i_u,k} + \sum_k t^c_{i_u,k}}\right)^{\delta_{E_{i_u}=\emptyset}} \left(\frac{1}{b + N_{i_u} + \sum_k t^c_{i_u,k}} \frac{S^{n_{i_u,k}+t^c_{i_u,k}+1}_{t^p_{i_u,k},a}}{S^{n_{i_u,k}+t^c_{i_u,k}}_{t^p_{i_u,k},a}}\right)^{\delta_{E_{i_u}\neq\emptyset}} \prod_{n=1}^{u-1} \frac{b + a\sum_k t^p_{i_n,k}}{b + N_u + \sum_k t^c_{i_n,k}} \frac{\gamma_{i_{n+1},i_n} + \sum_k s_{i_{n+1},i_n,k}}{\sum_l \gamma_{l,i_n} + \sum_k t^p_{i_n,k}} \ .$$

We need to sum this over all possible paths $i_1, i_2, ..., i_{u-1}$ starting at $i_1 = j$ in order to compute $p(\text{incr } n_{j,k} \,|\, \boldsymbol{n}_{1:J}, \boldsymbol{s}_{1:J,1:K}, \mathcal{H})$. The following recursive computation does this summation for a given j, k. Z^+_i is evaluated recursively when $E_i \neq \emptyset$ and Constraint (3) for node i at value k has equality. The recursive computation for Z^+_i is

$$\sum_{l \in E_i} Z^+_l \frac{b + a\sum_k t^p_{i,k}}{b + N_i + \sum_k t^c_{i,k}} \frac{\gamma_{l,i} + \sum_k s_{l,i,k}}{\sum_n \gamma_{n,i} + \sum_k t^p_{i,k}} \ .$$

Otherwise, Z^+_i is evaluated as

$$\begin{cases} \frac{1}{b+N_i+\sum_k t^c_{i,k}} \frac{S^{n_{i,k}+t^c_{i,k}+1}_{t^p_{i,k},a}}{S^{n_{i,k}+t^c_{i,k}}_{t^p_{i,k},a}} & \text{when } E_i \neq \emptyset \ , \\ \frac{\alpha_{i,k}+n_{i,k}+t^c_{i,k}}{\sum_k \alpha_{i,k}+\sum_k n_{i,k}+\sum_k t^c_{i,k}} & \text{when } E_i = \emptyset \ . \end{cases}$$

We evaluate $p(\text{incr } n_{j,k} \,|\, \boldsymbol{n}_{1:J}, \boldsymbol{s}_{1:J,1:K}, \mathcal{H}) = Z^+_j$, and this gives the proportionality for sampling which k to choose when incrementing a count in $\boldsymbol{n}_j$. Once k is sampled, then a path $i_1, i_2, ..., i_{u-1}$ in the constraint set should be sampled. This can be done using a similar function to the one just covered.

3.2.2 Decreasing a count

If $n_{j,k}$ is decreased by one, then Constraint (2) may be violated if the equality holds initially. This process is similar to the previous, except that now the $s_{i,j,k}$ are decreased and one needs to consider the set of ancestors of j reachable along paths where every element j' has equality for the Constraint (2) on the k-th value. Using a similar argument to previous, one gets a related recursive computation.

Z^-_i is evaluated recursively when $E_i \neq \emptyset$ and Constraint (2) for node i at value k has equality. The recursive computation for Z^-_i is

$$\sum_{l \in E_i} Z^-_l \frac{b + N_i + \sum_k t^c_{i,k} - 1}{b + a\sum_k t^p_{i,k} - 1} \frac{S^{n_{i,k}+t^c_{i,k}-1}_{t^p_{i,k}-1,a}}{S^{n_{i,k}+t^c_{i,k}}_{t^p_{i,k},a}} \frac{\sum_n \gamma_{n,i} + \sum_k t^p_{i,k} - 1}{\gamma_{l,i} + \sum_k s_{l,i,k} - 1} \frac{s_{l,i,k}}{t^p_{i,k}} \ .$$

Otherwise, Z^-_i is evaluated as

$$\begin{cases} \frac{\left(b+N_i+\sum_k t^c_{i,k}-1\right) S^{n_{i,k}+t^c_{i,k}-1}_{t^p_{i,k},a}}{S^{n_{i,k}+t^c_{i,k}}_{t^p_{i,k},a}} & \text{when } E_i \neq \emptyset \ , \\ \frac{\sum_k \alpha_{i,k}+\sum_k n_{i,k}+\sum_k t^c_{i,k}-1}{\alpha_{i,k}+n_{i,k}+t^c_{i,k}-1} & \text{when } E_i = \emptyset \ . \end{cases}$$

As before, $p(\text{decr } n_{j,k} \,|\, \boldsymbol{n}_{1:J}, \boldsymbol{s}_{1:J,1:K}, \mathcal{H}) = Z^-_j$. and this gives the proportionality for sampling which k to choose when wanting to decrement a count in $\boldsymbol{n}_j$. The path to choose when decrementing is chosen similarly.

3.2.3 Sampling Auxiliary Counts

The auxiliary counts $\boldsymbol{s}_{1:J,1:K}$ also need to be sampled. For correct Gibbs sampling, the space of moves must allow complete access to the state space of values $\boldsymbol{s}_{1:J,1:K}$ legal for a given $\boldsymbol{n}_{1:J}$. We introduce two move operators for sampling.

The first operator changes a single $s_{i,j,k}$ and possibly its ancestor counts. If Constraint (3) for node j at value k has equality, then $s_{i,j,k}$ must be zero and will not change. Allow sampling at node j for value k only when $n_{j,k}+t^c_{j,k} > 0$. By Constraint (2) for node j at value k, $s_{i,j,k}$ will be sampled keeping $s_{i,j,k} \leq n_{j,k} + t^c_{j,k} - \sum_{l \in E_j - \{i\}} s_{l,j,k}$ and keeping $t^p_{j,k} > 0$. Decreasing $s_{i,j,k}$ according to these constraints becomes

the same task as given in Section 3.2.2. Increasing $s_{i,j,k}$ within the constraints has no flow on affects so is the same as the standard incrementing formula in Section 3.2.1 where $u = 1$.

The second operator moves a count from one $s_{i,j,k}$ to another $s_{i',j,k}$. This can be put together by first doing a decrease as per Section 3.2.2 followed by an increase as per Section 3.2.1. The constraints at node j at value k will be unaffected by the combined move so no descendent auxiliary counts will change. A recursive argument shows these two operators make Gibbs sampling correct for the space of auxiliary counts $\boldsymbol{s}_{1:J,1:K}$.

3.3 Sampling the Variance Parameter b

In using these models for topic modelling, we have found performance is quite sensitive to the parameter b which controls the variance. For instance, for the distribution $\mathrm{PDP}\left(a, b, \mathrm{discrete}_K(\boldsymbol{\mu})\right)$ the hyperparameter b can thus roughly be thought of as the prior data count since variance is $O(1/(b+1))$.

We perform Gibbs sampling over b using auxiliary variables. First, consider the case where $a = 0$, discussed in (Teh et al., 2006). Consider the posterior for b, $p(\boldsymbol{n}_{1:J}, \boldsymbol{s}_{1:J,1:K} \,|\, \mathcal{H}, a = 0)$, proportional to

$$\prod_{j:E_j\neq\emptyset} \frac{b^{\sum_{i,k} s_{i,j,k}}\Gamma(b)}{\Gamma\left(b + N_j + \sum_{i,k} s_{j,i,k}\right)}$$

Introduce $q_j \sim \mathrm{Beta}\left(b, N_j + \sum_{i,k} s_{j,i,k}\right)$ as auxiliary variables. Then the joint posterior distribution for q_j and b is proportional to

$$b^{\sum_{i,j,k} s_{i,j,k}} \prod_{j:E_j\neq\emptyset} q_j^{b-1}(1-q_j)^{N_j+\sum_{i,k} s_{j,i,k}-1} .$$

The auxiliary sampling scheme then becomes:

$$q_j \sim \mathrm{Beta}\left(b, N_j + \sum_{i,k} s_{j,i,k}\right) \quad \text{for each } j,$$

$$b \sim \mathrm{Gamma}\left(\sum_{i,j,k} s_{i,j,k} + 1, \sum_j \log 1/q_j\right) .$$

For the case when $a > 0$ things become a bit more elaborate. Now the posterior is proportional to

$$\prod_{j:E_j\neq\emptyset} \frac{\Gamma(b)\Gamma\left(b/a + \sum_{i,k} s_{i,j,k}\right)}{\Gamma\left(b + N_j + \sum_{i,k} s_{j,i,k}\right)\Gamma(b/a)}$$

Introducing the same auxiliary variables as before yields a joint posterior distribution for q_j and b that is easily shown to be log concave, so the second step in the previous case ($a = 0$) is now replaced by an adaptive regression sampling step in b (Gilks and Wild, 1992).

4 Experiments

We extended standard LDA, shown in Figure 1 in two directions, which have been more fully developed and experimented with elsewhere (Du et al., 2010a; Du et al., 2010b). Here we cover basic results to demonstrate the effectiveness of the theory developed. The first model, the Segmented Topic Model (STM) (Du et al., 2010a) is shown in Figure 2, and allows a document to be broken into J segments. Each segment has its own topic proportions $\boldsymbol{\nu}_{i,j}$ which are related by our scheme using a PDP to the general proportions for the whole documents proportions $\boldsymbol{\mu}_i$. The second model, called the Sequential LDA (SeqLDA) (Du et al., 2010b), models the progressive topical dependencies among segments with a hidden Markov model of topic proportions $\boldsymbol{\nu}_{i,1}, ..., \boldsymbol{\nu}_{i,J}$, shown in Figure 3.

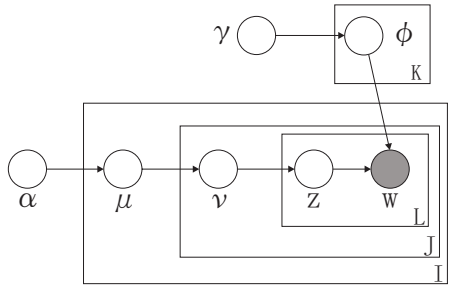


Figure 2: Segmented Topic Model.

A patent dataset was randomly selected from 5000 U.S. patents[2] granted between Jan. and

[2] All patents are from Cambia, `http://www.cambia.org/daisy/cambia/home.html`

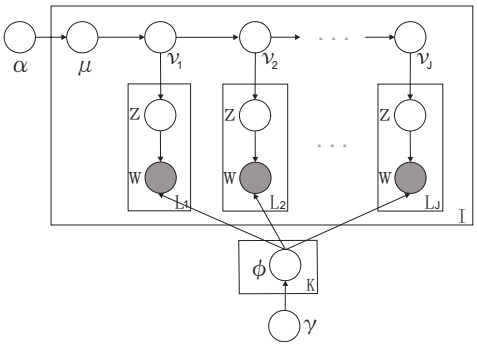


Figure 3: Sequential LDA.

Table 1: Perplexity on datasets.

	K	STM	LDA_D	LDA_S
G06-1000	100	1270	1712	1508
	150	1178	1595	1393
NIPS	100	1632	1991	2182
	150	1516	1881	2186

Mar. 2009 under the class "computing; calculating; counting" with international patent classification (IPC) code G06. Patents in this dataset called G06-1000 are split into paragraphs according to the original structure. All stop-words, extremely common words (*e.g.* top 40), and less common words (*i.e.* words appear less than 5 documents) have been removed. This leads to a vocabulary size of 10385 unique words segmented as 60,564 paragraphs, and 2,513,087 words. We also processed the NIPS dataset[3] removing bibliography material (everything after "References") and header material (everything before "Abstract") yielding 1,629 documents segmented as 174,474 sentences with 1,773,365 words total.

To evaluate the generalization capability of these models to unseen data, we compute perplexity, a standard measure in language modelling. The perplexity of a collection $\mathcal{D}$ of I documents is defined as $\exp\left\{-\frac{\sum_{i=1}^{I}\ln p(\boldsymbol{w}_i)}{\sum_{i=1}^{I}N_i}\right\}$ where $\boldsymbol{w}_i$ indicates all words in document i, and N_i indicates the total number of words in i. A lower perplexity over unseen documents means better generalization capability. In our experiments, it is computed based on the held-out method introduced in (Rosen-Zvi et al., 2004) with 80% for training and 20% for testing.

[3]It is available at `http://nips.djvuzone.org/txt.html`

Perplexity results appear in the table where LDA has been run twice, once on the full documents (LDA_D) and once on the segments within documents (LDA_S). Clearly, the STM model works well. The SeqLDA model was also run and not only gives better results than the LDA, but also reveals a strong sequential structure from segment to segment. To illustrate the sequential bahaviour of topics for SeqLDA, Figures 4 and 5 compare topic proportions for aligned topics for each segment (*i.e.* chapter) of the book *The Prince* by Machiavelli. SeqLDA is clearly seen to have topics flow better from one chapter to another than LDA. Refer to (Du et al., 2010a; Du et al., 2010b) for more detailed experimental results.

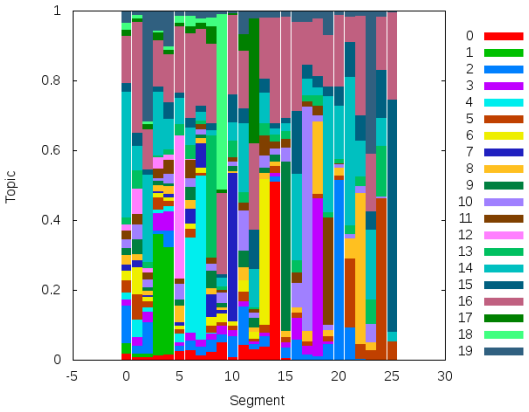


Figure 4: LDA topics in *The Prince*.

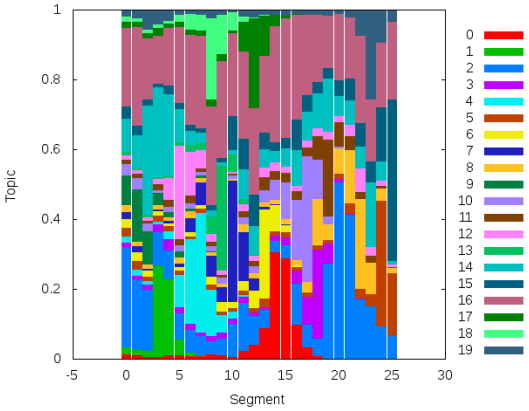


Figure 5: SeqLDA topics in *The Prince*.

5 Conclusion

We have shown how to perform inference on Bayesian networks of Dirichlet distributed probability vectors using Gibbs sampling over discrete auxiliary variables. Experiments demonstrate the general approach.

Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. This work was supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.

References

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

W. Buntine and M. Hutter. 2010. A Bayesian interpretation of the Poisson-Dirichlet process. *Available at: http://arxiv.org/abs/1007.0296v1.*

W.L. Buntine and A. Jakulin. 2006. Discrete components analysis. In *Subspace, Latent Structure and Feature Selection Techniques*. Springer-Verlag.

L. Du, W. Buntine, and H. Jin. 2010a. A segmented topic model based on the two-parameter Poisson-Dirichlet process. *Machine Learning (in press).*

L. Du, W. Buntine, and H. Jin. 2010b. Sequential latent Dirichlet allocation: Discover underlying topic structures within a document. Technical report, NICTA. In submission.

W.R. Gilks and P. Wild. 1992. Adaptive rejection sampling for Gibbs sampling. *Applied Statistics*, 41:337–348.

S. Goldwater, T. Griffiths, and M. Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In *NIPS 18*, pages 459–466. MIT Press.

R. Hanson, J. Stutz, and P. Cheeseman. 1991. Bayesian classification with correlation and inheritance. In *Proc. of the 12th IJCAI*, pages 692–698.

H. Ishwaran and L.F. James. 2001. Gibbs sampling methods for stick-breaking priors. *J. ASA*, 96(453):161–173.

M. Johnson, T.L. Griffiths, and S. Goldwater. 2007. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *NIPS 19*, pages 641–648. MIT Press.

S.L. Lauritzen. 1989. Mixed graphical association models. *Scand. Jnl. of Statistics*, 16(4):273–306.

D. Mochihashi and E. Sumita. 2008. The infinite Markov model. In *NIPS 20*, pages 1017–1024. MIT Press.

C.E. Rasmussen. 2000. The infinite Gaussian mixture model. In *NIPS 12*, pages 554–560. MIT Press.

M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. 2004. The author-topic model for authors and documents. In *Proc. of the 20th UAI*, pages 487–49.

E.B. Sudderth and M. Jordan. 2009. Shared segmentation of natural scenes using dependent Pitman-Yor processes. In *NIPS 21*. MIT Press.

Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. 2006. Hierarchical Dirichlet processes. *J. ASA*, 101.

Y.W. Teh. 2006a. A Bayesian interpretation of interpolated Kneser-Ney. Technical Report TRA2/06, School of Computing, National University of Singapore.

Y.W. Teh. 2006b. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. of the 21st ICCL and the 44th ACL*, pages 985–992.

A. Thomas, D.J. Spiegelhalter, and W.R. Gilks. 1992. BUGS: A program to perform Bayesian inference using Gibbs sampling. In *Bayesian Statistics 4*, pages 837–42. Clarendon Press.

H. Wallach, C. Sutton, and A. McCallum. 2008. Bayesian modeling of dependency trees using hierarchical Pitman-Yor priors. In *Proc. of the Workshop on Prior Knowledge for Text and Language (with ICML/UAI/COLT)*, pages 15–20.

F. Wood and Y. W. Teh. 2009. A hierarchical nonparametric Bayesian approach to statistical language model domain adaptation. In *Proc. of the Int. Conf. on Artificial Intelligence and Statistics*, volume 12.

F. Wood, C. Archambeau, J. Gasthaus, L.F. James, and Y.W. Teh. 2009. A stochastic memoizer for sequence data. In *Proc. of ICML'09.*

The Semantics of Intermediate CPTs in Variable Elimination

Cory J. Butz
University of Regina, Canada
butz@cs.uregina.ca

Wen Yan
University of Regina, Canada
yanwe111@cs.uregina.ca

Abstract

Variable elimination (VE), a central component of Bayesian network inference, starts and ends with clear structure and semantics, yet all intermediate distributions, whether normalized or unnormalized, are denoted as potentials. In this paper, a condition is given stating when intermediate distributions are defined with respect to the joint distribution. Theoretical and practical advantages of these new semantics are given.

1 Introduction

A *Bayesian network* (BN) (Pearl, 1988; Cowell et al., 1999; Jensen and Nielsen, 2007; Kjaerulff and Madsen, 2008) consists of a *directed acyclic graph* (DAG) and a corresponding set of *conditional probability tables* (CPTs). The independencies encoded in a DAG on variable set U indicate that the product of CPTs is a *joint probability distribution* $p(U)$. BN reasoning centres around eliminating variables. *Variable elimination* (VE) (Zhang and Poole, 1996), an inference algorithm for answering a query $p(X|E = e)$, repeatedly calls the *sum-out* (SO) algorithm to remove variables. SO removes a variable v as a two-step process. First, the product of all distributions involving v is taken. Second, v is marginalized out from the obtained product.

Koller and Friedman (2009) state that it is interesting to consider the semantics of the distribution output by SO when evidence is not considered. They mention that SO outputs a CPT $\phi(X|Y)$, but not necessarily with respect to $p(U)$. Butz et al. (2010) have shown a stronger result, namely, that every multiplication and every addition operation during VE's execution yields a CPT, albeit perhaps not with respect to $p(U)$.

In this paper, we address the semantics of VE's intermediate CPTs by providing a condition stipulating when they are defined with respect to $p(U)$. Roughly speaking, $\phi(X|Y)$ is $p(X|Y)$ provided there exists a topological ordering of all variables in the BN in which those variables used to build $\phi(X|Y)$ appear consecutively. In such cases, we say an intermediate CPT $\phi(X|Y)$ has a "p-label" and denote $\phi(X|Y)$ as $p(X|Y)$. It is important to observe that $\phi(X|Y)$ can be normalized or unnormalized, as well as involve evidence variables or not. It is noted that there are two kinds of paths that violate our condition, which, respectively, have temporary and permanent influences on SO. This work helps reveal structure and semantics in probabilistic reasoning with BNs, a worthy goal according to Pearl (1988) and Shafer (1996). We will also mention a practical advantage of this new semantic knowledge using the latest optimization techniques that are being applied in join tree propagation.

This paper is organized as follows. Section 2 contains background knowledge. The CPT structure of SO is discussed in Section 3. In Section 4, we establish semantics of VE's CPT structure. We extend the semantics to involve evidence in Section 5. Section 6 contains theoretical and practical advantages. Conclusions are given in Section 7.

Pp. 41–49 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

2 Background Knowledge

The following discussion draws mainly from Shafer (1996) and Olmsted (1983). Let $U = \{v_1, v_2, \ldots, v_n\}$ be a finite set of variables. Each v_i has a finite domain, denoted $dom(v_i)$. For a subset $X = \{v_i, \ldots, v_j\}$ of U, $dom(X)$ denotes the Cartesian product of the domains of the individual variables in X. Each element $x \in dom(X)$ is called a *configuration* of X.

A *potential* on $dom(X)$ is a function ψ such that $\psi(x) \geq 0$ for each $x \in dom(X)$, and at least one $\psi(x)$ is positive. A *joint probability distribution* on $dom(U)$ is a potential p on $dom(U)$ that sums to 1. A potential that sums to 1 is *normalized*; otherwise, it is *unnormalized*. We may write a set $\{v_1, v_2, \ldots, v_k\}$ as $v_1 v_2 \cdots v_k$ and use XY to denote $X \cup Y$. A *conditional probability table* (CPT) for X given disjoint Y, denoted $\phi(X|Y)$, is a potential on XY, satisfying the following condition: for each configuration $y \in dom(Y)$, $\sum_{x \in dom(X)} \phi(X = x \mid Y = y) = 1$. In writing $\phi(X|Y)$ with X and Y not disjoint, we always means $\phi(X|Y - X)$, and only configurations with non-zero probability are stored.

A discrete *Bayesian network* (BN) (Pearl, 1988) on $U = \{v_1, v_2, \ldots, v_n\}$ is a pair (B, C). B is a DAG with vertex set U. C is a set of CPTs $\{p(v_i|P_i) \mid i = 1, 2, \ldots, n\}$, where P_i denotes the parents (see below) of variable $v_i \in B$.

A *path* from v_1 to v_n is a sequence $v_1, v_2, \ldots, v_n$ with arcs (v_i, v_{i+1}), $i = 1, \ldots, n-1$ in B. With respect to a variable v_i, we define four sets: (i) the ancestors of v_i, denoted $A(v_i)$, are those variables having a path to v_i; (ii) the parents of v_i are those variables v_j such that arc (v_j, v_i) is in B; (iii) the descendants of v_i, denoted $D(v_i)$, are those variables to which v_i has a path; and, (iv) the children of v_i are those variables v_j such that arc (v_i, v_j) is in B. The ancestors of a set X of variables are defined as $A(X) = (\cup_{v_i \in X} A(v_i)) - X$. $D(X)$ is similarly defined. A *topological ordering* is an ordering $\prec$ of the variables in a BN B so that for every arc (v_i, v_j) in B, $v_i \prec v_j$. *Initial segments* of the ordering produce marginals of $p(U)$. A set W of variables in a DAG is an *initial segment* if the parents of each v_i in W are also in W.

Algorithm 1, called *sum-out* (SO), eliminates a single variable v from a set Φ of potentials, and returns the resulting set of potentials. The algorithm collect-relevant simply returns those potentials in Φ involving variable v.

Algorithm 1 sum-out(v,Φ)
begin
Ψ = collect-relevant(v,Φ)
ψ = the product of all potentials in Ψ
$\tau = \sum_v \psi$
Return $(\Phi - \Psi) \cup \{\tau\}$
end

Algorithm 2, called *variable elimination* (VE), computes $p(X \mid E = e)$ from a BN on U. VE calls SO to eliminate variables one by one. More specifically, in Algorithm 2, Φ is the set of CPTs in a BN, X is a list of query variables, E is a list of observed variables, e is the corresponding list of observed values, and σ is an elimination ordering for variables $U - XE$.

Algorithm 2 VE(Φ, X, E, e, σ)
begin
Set $E = e$ in all appropriate CPTs of Φ
While σ is not empty
  Remove the first variable v from σ
  Φ = sum-out(v, Φ)
$p(X, E = e)$ = the product of all $\phi \in \Phi$
$p(E = e) = \sum_X p(X, E = e)$
Return $p(X, E = e)/p(E = e)$
end

3 The CPT Structure of sum-out

Observe that VE starts and ends with clear structure and semantics, yet all intermediate distributions, whether normalized or unnormalized, are denoted as potentials. In their very comprehensive discussion, Koller and Friedman (2009) state that it is interesting to consider the semantics of the distribution constructed by summing out a variable from a BN not involving observed evidence. They point out that SO's marginalization step produces a CPT, but not necessarily with respect to the joint probability distribution $p(U)$.

Example 1. SO eliminates variable b from the BN in Figure 1 as:

$$\phi(c, e|a, d) = \sum_b p(b|a) \cdot p(c|b) \cdot p(e|b, d). \quad (1)$$

Thus, after marginalization, SO outputs a CPT.

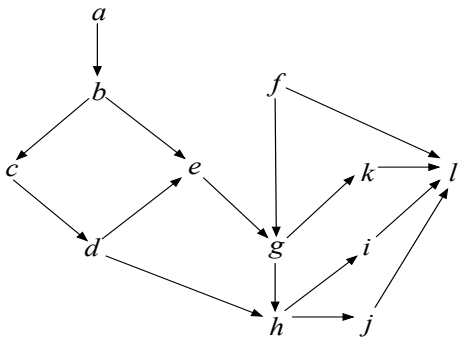


Figure 1: A Bayesian network.

Butz et al. (2010) have shown a stronger result, namely, every multiplication and every addition operation during VE's execution yields a CPT, albeit perhaps not with respect to $p(U)$.

Example 2. Each operation of (1) gives a CPT:

$$\begin{aligned} & \sum_b p(b|a) \cdot p(c|b) \cdot p(e|b, d) \\ = & \sum_b \phi(b, c|a) \cdot p(e|b, d) \quad (2) \\ = & \sum_b \phi(b, c, e|a, d) \quad (3) \\ = & \phi(c, e|a, d). \end{aligned}$$

Note that Shafer (1996) gives a condition under which each successive product of a sequence of CPTs always yields another CPT. In particular, for the case when multiplying just two CPTs $\phi(X_1|Y_1)$ and $\phi(X_2|Y_2)$, it is stated that

$$\phi(X_1X_2|Y_1Y_2) = \phi(X_1|Y_1) \cdot \phi(X_2|Y_2), \quad (4)$$

provided that X_2 is disjoint from X_1Y_1. The next example demonstrates, however, that (4) does not cover all possible cases encountered when applying SO on a BN.

Example 3. Consider eliminating variable c from the following BN:

$$p(a) \cdot p(b) \cdot p(c) \cdot p(d|a, b, c) \cdot p(e|c, d).$$

By Algorithm 1, we obtain

$$\sum_c p(c) \cdot p(d|a, b, c) \cdot p(e|c, d).$$

The optimal multiplication ordering, assuming binary variables and positive probabilities, is

$$\sum_c (p(c) \cdot p(e|c, d)) \cdot p(d|a, b, c).$$

By (4), $p(c) \cdot p(e|c, d)$ is $\phi(c, e|d)$, giving

$$\sum_c \phi(c, e|d) \cdot p(d|a, b, c). \quad (5)$$

As (4) no longer applies, the product of the two CPTs in (5) must be denoted as a potential:

$$\sum_c \psi(a, b, c, d, e).$$

While (Butz et al., 2010) reveals the CPT structure of $\psi(a, b, c, d, e)$ as $\phi(c, d, e|a, b)$, the remaining unanswered question is semantic. Is $\phi(c, d, e|a, b)$ equal to $p(c, d, e|a, b)$?

4 Semantics Without Evidence

We first consider eliminating variables without observed evidence. It can easily be shown that SO's marginalization operation will always yield a CPT with the same kind of label as the CPT being marginalized. Thus, we focus on the semantics of multiplication.

Theorem 1. *Given a BN B on U and $X \subseteq U$. Then*

$$p(X|Y) = \prod_{v_i \in X} p(v_i|P_i),$$

if there is a topological ordering $\prec$ of B in which the variables in X appear consecutively, where $Y = (\cup_{v_i \in X} P_i) - X$.

Proof. Suppose there exists a topological ordering $\prec$ of the variables in B in which the variables in X appear consecutively. Let W be the set of all variables appearing in $\prec$ before any variable in X. By (Shafer, 1996), the variables in W and WX are both initial segments, meaning that

$$p(W) = \prod_{v_w \in W} p(v_w|P_w) \quad (6)$$

and

$$p(WX) = \prod_{v_w \in W} p(v_w|P_w) \cdot \prod_{v_x \in X} p(v_x|P_x). \quad (7)$$

By substitution of (6) into (7),

$$p(WX) \;=\; p(W) \cdot \prod_{v_x \in X} p(v_x|P_x).$$

By (Butz et al., 2010),

$$p(WX) \;=\; p(W) \cdot \phi(X|Y). \quad (8)$$

According to $\prec$, the variables Y must be contained in W, by (Shafer, 1996). Let $V = W - Y$, so $W = VY$. Then (8) can be rewritten as

$$p(VYX) \;=\; p(VY) \cdot \phi(X|Y).$$

Marginalizing away V yields

$$p(YX) \;=\; p(Y) \cdot \phi(X|Y).$$

By rearrangement, we obtain our desired result

$$p(X|Y) \;=\; \phi(X|Y).$$

□

Example 4. By Theorem 1, $\phi(b, c|a)$ in (2) is $p(b, c|a)$, since b and c can appear consecutively in a topological order $\prec$ of B in Figure 1. However, $\phi(b, c, e|a, d)$ in (3) is not guaranteed to be $p(b, c, e|a, d)$, since every topological order $\prec$ has d between c and e, i.e., $c \prec d \prec e$.

Our topological condition is sufficient but not necessary to ensure CPTs with p-labels.

Example 5. Consider eliminating b from the BN in Figure 1. Suppose the CPTs for $a, \ldots, e$ are defined such that their marginal has only one configuration with a non-zero probability, say, $p(a = 0, b = 0, c = 0, d = 0, e = 0)$ is 1. In this extreme case, it can be verified that

$$p(b, c, e|a, d) = p(b|a) \cdot p(c|b) \cdot p(e|b, d).$$

To ensure p-label CPTs for any BN instance B, we extend SO as *sum-out-as-p* (SOP), which calls *collect-topological* (CT) to collect any CPT needed to satisfy our topological ordering requirement in B.

Algorithm 3 sum-out-as-p(v,Φ)
begin
Ψ = collect-relevant(v, Φ)
Θ = collect-topological(Ψ, Φ)
$p(X|Y)$ is the product of all CPTs in Ψ and Θ
$p(X - v|Y) = \sum_v p(X|Y)$
Return $(\Phi - \Psi - \Theta) \cup \{p(X - v|Y)\}$
end

Algorithm 4 collect-topological(Ψ,Φ)
begin
X is the union of all X_i where $p(X_i|Y_i) \in \Psi$
Let Z be $A(X) \cap D(X)$
Let Ω be those $p(X_i|Y_i)$ in Φ with $X_i \cap Z \neq \emptyset$
Return Ω
end

Example 6. Consider how SOP eliminates h, b and f from Figure 1. For h, $\Psi = \{p(h|d, g), p(i|h), p(j|h)\}$. CT returns $\Omega = \emptyset$, as $X = hij$, $Z = A(hij) \cap D(hij) = \emptyset$. In SOP, $\Theta = \emptyset$, so

$$p(i, j|d, g) = \sum_h p(h|d, g) \cdot p(i|h) \cdot p(j|h).$$

For b, $\Psi = \{p(b|a), p(c|b), p(e|b, d)\}$ and, as $Z = A(bce) \cap D(bce) = d$, $\Theta = \{p(d|c)\}$. Thus,

$$p(c, d, e|a) = \sum_b p(b|a) \cdot p(c|b) \cdot p(d|c) \cdot p(e|b, d).$$

For f, $\Psi = \{p(f), p(g|e, f), p(l|f, k, i, j)\}$ and, since $Z = A(fgl) \cap D(fgl) = hijk$, we have $\Theta = \{p(i, j|d, g), p(k|g)\}$. Therefore,

$$\sum_f p(f)\ p(g|e, f)\ p(i, j|d, g)\ p(k|g)\ p(l|f, k, i, j)$$
$$= p(g, i, j, k, l\ |d, e).$$

Observe that every elimination in Example 6 yielded a p-label CPT. Also note that Z in the collect-topological algorithm can be quickly obtained from the transitive closure of a DAG, which can be found in $O(n^3)$ time (Cormen et al., 2009). It must be made clear that we are not advocating that SOP be considered as a new approach to inference. Instead, SOP can shed insight into the semantics of SO's intermediate CPTs. SO is ensured to yield a CPT with a p-label, if it collects the same CPTs as SOP does.

Example 7. Recall Example 6. When eliminating variable h, SO will yield a p-label, since SO and SOP collect the same CPTs. On the contrary, to eliminate variable b, SO can compute the following ϕ-label:

$$\phi(b, c, e|a, d) = p(b|a) \cdot p(c|b) \cdot p(e|b, d).$$

A p-label is not necessarily obtained here due to the fact that SOP also collects $\Theta = \{p(d|c)\}$.

More generally, every product taken in SO of two CPTs $\phi(X_1|Y_1)$ and $\phi(X_2|Y_2)$ will be $p(X_1X_2|Y_1Y_2)$, provided the topological requirement is met. Let us focus on eliminating a single variable from a BN. There are two kinds of paths warranting attention. The first only involves children of the variable being eliminated. When eliminating a variable v from a BN, the CPTs of v's children must be multiplied in an order consistent with some topological ordering of the DAG. The following example illustrates how this condition has a temporary influence within SO, meaning that intermediate CPTs can alternate between ϕ- and p-labels.

Example 8. Consider the elimination of variable b from the BN in Figure 2:

$$\begin{aligned}
& \sum_b p(b) \cdot p(e|b, d) \cdot p(d|a, b) \cdots && (9)\\
= & \sum_b \phi(b, e|d) \cdot p(d|a, b) \cdot p(g|b, f) \cdots && (10)\\
= & \sum_b p(b, d, e|a) \cdot p(g|b, f) \cdot p(f|b, c, e) \cdots && (11)\\
= & \sum_b \phi(b, d, e, g|a, f) \cdot p(f|b, c, e) \cdots && (12)\\
= & \sum_b p(b, d, e, f, g|a, c) \cdot p(i|b, g, h) \cdots && (13)\\
= & \sum_b \phi(b, d, e, f, g, i|a, c, h) \cdot p(j|b, i) && (14)\\
= & \sum_b \phi(b, d, e, f, g, i, j|a, c, h) && (15)\\
= & \phi(d, e, f, g, i, j|a, c, h). && (16)
\end{aligned}$$

Example 8 demonstrates how the intermediate CPTs can alternate between having and not having p-labels. A ϕ-label can be obtained when multiplying the CPTs for b and e in (10), since there is a path from b to e going through d (so $b \prec d \prec e$). Since the CPT $p(d|a, b)$ for d has been collected by SO, a p-label can be subsequently re-obtained, as shown in (11). Similar remarks hold for multiplying this product with the CPT for g before that for f, as shown in (12) - (13).

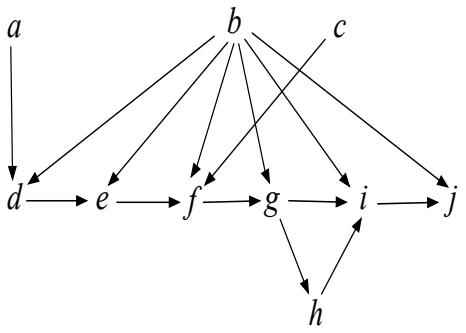


Figure 2: Illustrating the alternating pattern of intermediate CPTs with p-labels in Example 8.

The second kind of path, however, has a permanent influence on the semantics of SO's intermediate CPTs; it involves variables that are not children of the variable being eliminated. Recall Example 8 where variable b is being eliminated and consider (13) - (14). Once the CPT $p(i|b, g, h)$ is multiplied, all CPTs subsequently constructed by SO during the elimination of b can have a ϕ-label as in (14) - (16). The reason is that there is a path from b to i going through h (so $b \prec g \prec h \prec i$). However, the CPT $p(h|g)$ is not collected by SO as $p(h|g)$ does not involve b. Hence, the only way to ensure a subsequent p-label is to wait for $p(h|g)$ to be multiplied during a different call to SO, say to eliminate h.

Theorem 2. *Given a BN B on U, let $\phi(X|Y)$ be any CPT that VE computes by multiplication. Then $\phi(X|Y)$ is $p(X|Y)$, if there is a topological ordering $\prec$ of B in which the variables in DX appear consecutively, where D are those variables that were eliminated by SO in building $\phi(X|Y)$.*

The proof of Theorem 2 is similar to that of Theorem 1 and will be shown in a separate manuscript.

Example 9. Continuing from Example 8, consider the elimination of variable h.

$$\begin{aligned} & \sum_h p(h|g) \cdot \phi(d,e,f,g,i,j|a,c,h) \\ = & \sum_h p(d,e,f,g,h,i,j|a,c) \qquad (17) \\ = & \; p(d,e,f,g,i,j|a,c). \end{aligned}$$

In Example 9, a p-label is obtained in (17) as there exists a topological ordering $\prec$ of the BN in Figure 2 where the variables in b,d,e,f,g,h,i,j appear consecutively.

5 Semantics With Evidence

Suppose we observe the values e of a set E of variables contained in U. Before a disjoint set D also contained in U is eliminated from the BN, those CPTs containing evidence variables are modified by multiplying them with evidence potentials. An *evidence potential*, denoted $1(E)$, assigns probability 1 to the single configuration e of E and probability 0 to all other configurations of E. If a BN CPT $p(v_i|P_i)$ contains at least one evidence variable, then 1_i denotes $1(E)$ restricted to those evidence variables appearing in $p(v_i|P_i)$. Hence, the product $p(v_i|P_i) \cdot 1_i$ keeps the configurations agreeing with $E = e$ while deleting the rest. By $1(\emptyset)$, we denote 1.

Intermediate potentials, constructed during inference by VE, can always have their conditional probabilities (Theorem 3) and semantics (Corollary 1) identified.

Theorem 3. *Given a BN B and evidence $E = e$, every VE distribution constructed by multiplication can be expressed as the product of a CPT and an evidence potential.*

Proof. Let the constructed distribution be $\phi_1 \cdot \phi_2$. We can always equivalently rewrite ϕ_1 as the marginalization of the product of the CPTs and evidence potentials used to build ϕ_1:

$$\phi_1 = \sum_{D_1} \prod_{i=1}^{j} (p(v_i|P_i) \cdot 1_i) \cdot \prod_{i=j+1}^{k} p(v_i|P_i),$$

where D_1 is the set of variables marginalized away by SO from the product of k CPTs in B, and where j evidence potentials were used. Since $E \cap D_1 = \emptyset$, we have

$$\begin{aligned} \phi_1 &= \prod_{i=1}^{j} 1_i \cdot \sum_{D_1} \prod_{i=1}^{j} p(v_i|P_i) \cdot \prod_{i=j+1}^{k} p(v_i|P_i) \\ &= \prod_{i=1}^{j} 1_i \cdot \sum_{D_1} \prod_{i=1}^{k} p(v_i|P_i). \end{aligned}$$

Similarly, for ϕ_2,

$$\begin{aligned} \phi_2 &= \sum_{D_2} \prod_{i=k+1}^{l} (p(v_i|P_i) \cdot 1_i) \cdot \prod_{i=l+1}^{m} p(v_i|P_i) \\ &= \prod_{i=k+1}^{l} 1_i \cdot \sum_{D_2} \prod_{i=k+1}^{l} p(v_i|P_i) \cdot \prod_{i=l+1}^{m} p(v_i|P_i) \\ &= \prod_{i=k+1}^{l} 1_i \cdot \sum_{D_2} \prod_{i=k+1}^{m} p(v_i|P_i). \end{aligned}$$

Thus, the product $\phi_1 \cdot \phi_2$ is

$$\prod_{i=1}^{j} 1_i \cdot \sum_{D_1} \prod_{i=1}^{k} p(v_i|P_i) \cdot \prod_{i=k+1}^{l} 1_i \cdot \sum_{D_2} \prod_{i=k+1}^{m} p(v_i|P_i).$$

By SO, D_2 and D_1 have no common variables with the CPTs $p(v_i|P_i)$, $i = 1,\ldots,k$ and $i = k+1,\ldots,m$, respectively. Therefore, we have:

$$\prod_{i=1}^{j} 1_i \cdot \prod_{i=k+1}^{l} 1_i \cdot \sum_{D_1 D_2} \prod_{i=1}^{k} p(v_i|P_i) \cdot \prod_{i=k+1}^{m} p(v_i|P_i).$$

Rearranging yields

$$\prod_{i=1}^{j} 1_i \cdot \prod_{i=k+1}^{l} 1_i \cdot \sum_{D_1 D_2} \prod_{i=1}^{m} p(v_i|P_i).$$

Let $\phi(X|Z) = \prod_{i=1}^{m} p(v_i|P_i)$. Then we have

$$\prod_{i=1}^{j} 1_i \cdot \prod_{i=k+1}^{l} 1_i \cdot \sum_{D_1 D_2} \phi(X|Z).$$

As $D_1 D_2 \subseteq X$ (Butz et al., 2010), let $W = X - D_1 D_2$. Thus, the multiplication $\phi_1 \cdot \phi_2$ is

$$1(E \cap WZ) \cdot \phi(W|Z),$$

where the product of the evidence potentials is $1(E \cap WZ)$. □

Corollary 1. *In the proof of Theorem 3, $\phi(W|Z)$ is $p(W|Z)$, provided there is a topological ordering $\prec$ of B in which the variables in DW appear consecutively, where D are those variables that were eliminated to build $\phi(W|Z)$.*

6 Advantages

We stress the improvement in clarity and suggest a direction of practical investigation.

Kjaerulff and Madsen (2008) suggest that in working with probabilistic networks it is convenient to denote distributions as potentials. Similarly, Koller and Friedman (2009) would denote the start of Example 8 as

$$\sum_b \psi_2(b) \cdot \psi_5(e,b,d) \cdot \psi_4(d,a,b) \cdots .$$

Observe that both the p-labels and the CPT structure have been destroyed even before the distributions in memory have been modified. It is then more meaningful to keep the CPT structure and semantics highlighted in (9) - (16).

Consider evidence $i = 1$ and $h = 0$ in the BN in Figure 3, which Koller and Friedman (2009) call non-trivial. All intermediate distributions are denoted as potentials in the computation of $p(j \mid i = 1, h = 0)$. However, it follows from Theorem 3 and Corollary 1 that structure and semantics can still be identified.

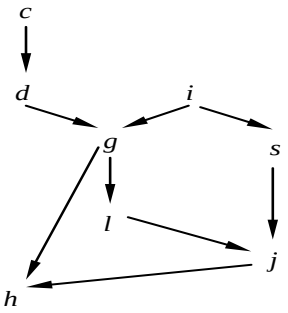


Figure 3: The DAG of a non-trivial BN.

Example 10. Computing $p(j \mid i = 1, h = 0)$ in the BN of Figure 3 involves, in part,

$$\sum_{c,d,g,s,l} p(c) \cdot p(d|c) \cdot p(i) \cdot p(s|i) \cdot p(g|d,i) \cdot p(l|g) \cdot p(j|l,s) \cdot p(h|g,j) \cdot 1(i=1) \cdot 1(h=0).$$

Eliminating variables c and d requires

$$\sum_d p(g|d,i) \cdot 1(i=1) \cdot \sum_c p(c) \cdot p(d|c) \quad (18)$$

$$= \sum_d p(g|d,i=1) \cdot \sum_c p(c,d) \quad (19)$$

$$= \sum_d p(g|d,i=1) \cdot p(d) \quad (20)$$

$$= \sum_d p(d,g|i=1)$$

$$= p(g|i=1).$$

Variable g can be eliminated as:

$$\sum_g p(g|i=1) \cdot p(l|g) \cdot p(h|g,j) \cdot 1(h=0)$$

$$\sum_g p(g|i=1) \cdot p(l|g) \cdot p(h=0|g,j)$$

$$= \sum_g p(g,l|i=1) \cdot p(h=0|g,j) \quad (21)$$

$$= \sum_g \phi(g,l,h=0|i=1,j) \quad (22)$$

$$= \phi(l,h=0|i=1,j). \quad (23)$$

The remainder of the example is omitted.

Example 10 shows that all intermediate distributions have structure and semantics, regardless of: the involvement of evidence potentials (18); the side or sides of the bar on which evidence appears (21), (22); marginalization operations (19); and p-labels (20) or ϕ-labels (23). Now let us turn to efficiency issues.

All previous join tree propagation algorithms either exclusively apply VE or *arc reversal* (AR) (Olmsted, 1983) at all join tree nodes (Madsen, 2010), or pick whether to apply VE or AR at each node (Butz et al., 2009a). A practical advantage of our new semantics is the ability to construct messages using both VE and AR at the *same* join tree node.

Example 11. Consider a join tree with three nodes $abcdefgh$, $fgij$ and fik. The CPTs assigned to $abcdefgh$ are $p(a)$, $p(b|a)$, $p(c|b)$, $p(d|c)$, $p(e|b,d)$, $p(f|e)$, $p(g|e)$, $p(h|a,b,c,d,e,f,g)$, while $fgij$ is provided $p(i|g)$ and $p(j|f,g,i)$, and fik is given $p(k|f,i)$. Butz et al.'s (2009b) message identification process indicates that $abcdefgh$ will pass $p(f)$ and $p(g|f)$ to node $fgij$, which, in turn, will pass $p(f)$ and $p(i|f)$ to fik. Madsen (2010) and Butz et al. (2009a) would apply AR at node $abcdefgh$, i.e., with some abuse of notation:

$$p(f) \cdot p(g|f) \quad = \quad \sum_{a,b,c,d,e}^{AR} p(a) \cdots p(f|e) \cdot p(g|e),$$

where h is removed as a barren variable. AR must be applied to remove the last variable e. However, by examining the semantics of VE, it can be verified that the elimination of variables a, b, c and d gives $p(e)$. Thus, apply VE to eliminate variables a, b, c and d, and then apply AR to eliminate variable e:

$$\begin{aligned}
&\sum_{e}^{AR}\sum_{a,b,c,d}^{VE} p(a)\cdot p(b|a)\cdot p(c|b)\cdot p(d|c)\cdots p(g|e)\\
=\;&\sum_{e}^{AR}\sum_{b,c,d}^{VE} p(b)\cdot p(c|b)\cdot p(d|c)\cdot p(e|b,d)\cdots p(g|e)\\
=\;&\sum_{e}^{AR}\sum_{c,d}^{VE} \phi(c,e|d)\cdot p(d|c)\cdot p(f|e)\cdot p(g|e)\\
=\;&\sum_{e}^{AR}\sum_{d}^{VE} p(d,e)\cdot p(f|e)\cdot p(g|e)\\
=\;&\sum_{e}^{AR} p(e)\cdot p(f|e)\cdot p(g|e).
\end{aligned}$$

It can be verified that applying AR at $abcdefgh$ requires more computation than applying the combination of VE and AR as shown above.

Empirical results will be reported separately.

7 Conclusions

Pearl (1988) emphasizes that probabilistic reasoning is not about numbers and is instead about the structure of reasoning. Our work here ascribes semantics to the intermediate CPTs of VE. This is the primary contribution of this paper. A practical advantage of these semantics was illustrated using the latest optimization techniques employed in join tree propagation and requires further study.

Intermediate CPTs constructed by VE could be labeled solely with p-labels, provided the label of each distribution is an expression rather than a single term. That is, label each CPT output by SO as a fraction, where the numerator is the p-label CPT output by SOP and the denominator is the factorization of "missing" CPTs in Θ. Thus, whereas SO can eliminate variable b in Example 7 as $\phi(c,e|a,d)$, it may be semantically more meaningful to take another step and label it $p(c,d,e|a)/p(d|c)$.

Acknowledgments

This research is supported by NSERC Discovery Grant 238880. The authors thank K. Williams for useful comments.

References

C.J. Butz, K. Konkel and P. Lingras. 2009a. Join tree propagation utilizing both arc reversal and variable elimination. In *Twenty Second International Florida Artificial Intelligence Research Society Conference*, pages 523–528.

C.J. Butz, H. Yao and S. Hua. 2009b. A join tree probability propagation architecture for semantic modeling, *Journal of Intelligent Information Systems*, 33(2):145-178.

C.J. Butz, W. Yan, P. Lingras and Y.Y. Yao. 2010. The CPT structure of variable elimination in discrete Bayesian networks. Advances in Intelligent Information Systems. SCI 265. Z.W. Ras and L.S. Tsay (Eds.). Springer, pages 245-257.

T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein. 2009. *Introduction to Algorithms*. MIT Press.

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

F.V. Jensen and T.D. Nielsen. 2007. *Bayesian Networks and Decision Graphs*. Springer.

U.B. Kjaerulff and A.L. Madsen. 2008. *Bayesian Networks and Influence Diagrams*. Springer.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles And Techniques*. MIT Press.

A.L. Madsen. 2010. Improvements to message computation in Lazy propagation. *Int. J. Approx. Reason*, 51(5):499-514.

S. Olmsted. 1983. On representing and solving decision problems, Ph.D. Thesis, Department of Engineering Economic Systems, Stanford University, Stanford, California.

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

G. Shafer. 1996. *Probabilistic Expert Systems*. SIAM.

N.L. Zhang and D. Poole. 1996. Exploiting causal independence in Bayesian network inference, *J. Artif. Intell. Res.*, (5):301-328.

Learning Recursive Probability Trees from Probabilistic Potentials

Andrés Cano, Manuel Gómez-Olmedo, Serafín Moral, Cora B. Pérez-Ariza
Department of Computer Science and Artificial Intelligence
University of Granada, Spain
{acu,mgomez,smc,cora}@decsai.ugr.es

Antonio Salmerón
Department of Statistics and Applied Mathematics
University of Almería, Spain
antonio.salmeron@ual.es

Abstract

A recursive probability tree (RPT) is an incipient data structure for representing the distributions in a probabilistic graphical model. RPTs capture most of the types of independencies found in a probability distribution. The explicit representation of these features using RPTs simplifies computations during inference. This paper describes a learning algorithm that builds a RPT from a probability distribution. Experiments prove that this algorithm generates a good approximation of the original distribution, thus making available all the advantages provided by RPTs.

1 Introduction

The required size for representing probability distributions in a probabilistic graphical model (like a Bayesian network) is exponential in the number of variables. A Bayesian network is an efficient representation of a joint probability distribution, since it exploits independencies among the variables, but it cannot directly represent *context-specific independencies* (Boutilier et al., 1996) within the distributions. Probability trees have been previously used to represent this kind of independencies within probability potentials (Cano et al., 2000). Moreover, sometimes a probability distribution can be obtained through the multiplication (factorization) of a list of smaller distributions by detecting proportionalities within it (Martínez et al., 2002; Martínez et al., 2005; Martínez et al., 2006). Recently, a new data structure for representing potentials was introduced (Cano et al., 2009): *Recursive Probability Trees (RPTs)*. This kind of tree is a generalization of a probability tree. It allows to represent context-specific independencies within distributions, while keeping potentials factorized.

Like probabilistic decision graphs (Jaeger, 2004) and chain event graphs (Smith and Anderson, 2008), RPTs can be used as a standalone representation of joint probability distributions, and probabilistic inference can be fully carried out using this single structure, as the necessary operations, namely product, marginalization and restriction, are well defined over this data structure (Cano et al., 2009).

This paper presents a learning algorithm able to decompose a probability distribution into smaller pieces, by detecting context-specific independencies and multiplicative decompositions. This decomposition will be represented as a RPT. The rest of the paper is organized as follows: Section 2 defines RPTs and describes their features; Section 3 presents the algorithm used for constructing a RPT from a probabilistic potential; Section 4 shows the experiments performed for testing the performance of the algorithm; and finally Section 5 presents conclusions as well as future research directions.

Pp. 49–57 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

2 Recursive Probability Trees

A Recursive Probability Tree (Cano et al., 2009) (hereafter referred to as RPT) is a directed tree with two different kinds of inner nodes (Split nodes and List nodes), and two types of leaf nodes (Value nodes and Potential nodes). A Split node represents a discrete variable. A List node represents a multiplicative factorization by listing all the factors to which a potential is decomposed. It contains one outgoing arc for every factor in the decomposition. A Value node represents a non-negative real number. Finally, a Potential node stores a full potential internally using an arbitrary representation. Fig. 1 shows a RPT (left part) and the Bayesian network whose joint probability distribution is encoded in the tree (right part). Note how the potentials are enclosed in Potential nodes, and how the List node represents a multiplicative factorization. Using this structure, it is possible to represent context-specific independencies within a probability distribution, as shown in Fig. 2, as well as factorizations (involving the whole potential or parts of it). Proportionalities within the probability distribution are also easily represented using this structure (see Fig. 2).

In (Cano et al., 2009) we give a formal definition of a RPT and a method to obtain the value of the potential for each configuration of its variables.

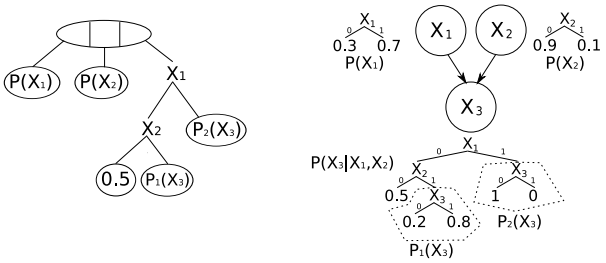


Figure 1: RPT (left) encoding of a Bayesian network distribution (right)

3 Constructing a RPT from a probabilistic potential

In this section we shall describe our proposal for transforming any given probabilistic potential (for instance, a probability table) into a RPT. Our proposal is aimed at detecting context specific independencies and multiplicative factorizations present in a probabilistic potential. In order to detect context specific independencies, we follow an approach similar to the procedure used for constructing probability trees (Salmerón et al., 2000), which is based on selecting variables for Split nodes according to their information gain, in a similar way as variables are selected when constructing decision trees (Quinlan, 1986). Regarding the multiplicative decomposition, the basic idea is to make groups of variables using their mutual information as a selection criterion. The groups are later used to obtain the potentials that make up the multiplicative decomposition.

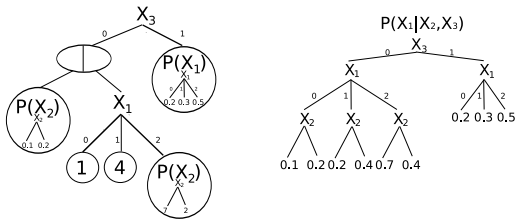


Figure 2: Distribution with context specific independencies and proportional values (right), and the corresponding RPT (left)

The starting point is a potential f defined over a set of variables $\mathbf{X}$, and the aim of the algorithm is therefore to find a representation of f as a RPT. We denote $S = \sum_{\mathbf{x}} f(\mathbf{x})$ as the sum of all the values in f and $S(Y = y) = \sum_{\mathbf{z}} f(\mathbf{z}, y)$ (sum of values consistent with $Y = y$). The procedure we propose operates over an auxiliary graph structure G_f with vertex set $\mathbf{X}$ where two variables $Y, Z \in \mathbf{X}$ will be linked if there is a probabilistic dependence between them. More precisely, a link $Y - Z$ is present in G_f if the dependence between Y and Z exceeds a given threshold $\varepsilon > 0$. We use the mutual information as a measure of dependence, and hence, a link $Y - Z$ will be included only if

$$I(Y, Z) = \sum_{y,z} p(y, z) \log \frac{p(y, z)}{p(z)p(y)} > \varepsilon, \qquad (1)$$

where $p(y, z) = \frac{f^{\downarrow YZ}(y,z)}{S}$ and $f^{\downarrow YZ}$ is the marginal of f over variables Y and Z. Each

link $Y - Z$ is weighted with $I(Y, Z)$. After constructing graph G_f, a first factorization of f is readily obtained if G_f is disconnected in n connected components $\mathbf{X}_1 \cup \cdots \cup \mathbf{X}_n = \mathbf{X}$. The factorization is given by

$$f(\mathbf{x}) = f_1(\mathbf{x}_1) \cdots f_n(\mathbf{x}_n) S_n, \tag{2}$$

where $f_i = f^{\downarrow \mathbf{X}_i}$, $i = 1, \ldots, n$, and S_n is a normalization factor required to keep the sum of $f_1(\mathbf{x}_1) \cdots f_n(\mathbf{x}_n)$ equal to the sum of $f(\mathbf{x})$:

$$S_n = \frac{\sum_{\mathbf{x}} f(\mathbf{x})}{\sum_{\mathbf{x}} (f_1(\mathbf{x}_1) \cdots f_n(\mathbf{x}_n))}. \tag{3}$$

Hence, potential f can be represented as a RPT where the root node would be a List node containing the factors in Eq. (2). On the contrary, if graph G_f remains as a single connected component, it means that the potential is not decomposable as a list of factors with disjoint variables. However, conditional decompositions are possible. In order to seek for such context specific factorizations we must compute for each variable $Y \in \mathbf{X}$ the following value

$$V(Y) = \sum_{Z \text{ neighbour of } Y} I(Y, Z). \tag{4}$$

Next, we choose Y_0 such that $Y_0 = \arg\max_{Y \in \mathbf{X}} V(Y)$. This heuristic removes the variable more dependent on the remaining variables in the graph. This way we can split the graph into several connected components representing independent parts within the potential analyzed.

The procedure described above is repeated, but restricted to variable Y_0. That is, we construct a graph $G_f^{Y_0}$ by removing Y_0 and its links from G_f and re-weighting each remaining link $Z - U$ in $G_f^{Y_0}$ with

$$I(Z, U|Y_0) = \sum_{z,u,y_0} p(z, u, y_0) \log \frac{p(z, u|y_0)}{p(z|y_0)p(u|y_0)},$$

where

$$p(z, u, y_0) = \frac{f^{\downarrow ZUY_0}(z, u, y_0)}{\sum_{z,u,y_0} f^{\downarrow ZUY_0}(z, u, y_0)}$$

and the conditional distributions $p(z, u|y_0)$, $p(z|y_0)$ and $p(u|y_0)$ are computed from $p(z, u, y_0)$. If the value of the marginal for some configuration of the conditioning variable is equal to zero, then the conditional mutual information reduces to a summation of terms of the form $0log0$, and assumed to be zero. Again, we consider a threshold $\varepsilon > 0$ so that only those links $Z - U$ where $I(Z, U \mid Y_0) > \varepsilon$ will be added to the graph, and weighted as $I(Z, U \mid Y_0)$.

As a previous step to the creation of $G_f^{Y_i}$, we must sort Y_i into set $\mathbf{Y}_1$ if the variable was connected to all the other variables in the connected component before being removed; otherwise, the variable will be appended to set $\mathbf{Y}_2$. These two sets will help us to discern between scenarios with context-specific independencies and factorizations.

In general, if $Y_0, \ldots, Y_m$ have been chosen, everything must be turned conditional on $Y_1, \ldots, Y_m$ before selecting Y_{m+1}. The process stops when a division of the graph is found or when there is no variable left to choose, i.e. the graph has two variables and both are connected.

In the second case (the graph has only two connected variables) there is no further possible decomposition. In the first case, suppose that after choosing $\mathbf{Y} = \{Y_0, ..., Y_m\}$ the graph is decomposed into n connected components $\mathbf{Z} = \mathbf{Z}_1 \cup \cdots \cup \mathbf{Z}_n$, with $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$.

Next, a RPT is built with the variables in $\mathbf{Y}_1$ (containing Split nodes). For each leaf h of this tree, suppose that $\mathbf{Y}_1 = \mathbf{y}_1$ is the assignment compatible with h, then potential f^h is stored in leaf h, where f^h is defined as $f^h = f^{R(\mathbf{Y}_1 = \mathbf{y}_1)}$, and $f^{R(\mathbf{Y}_1 = \mathbf{y}_1)}$ denotes the potential f restricted to assignment $(\mathbf{Y}_1 = \mathbf{y}_1)$. Potential f^h is decomposed as

$$f^h(\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{y}_2) := S^h \prod_{i=1}^{n} f^h(\mathbf{z}_i, \mathbf{y}_2),$$

$$S^h = \frac{\sum_{\mathbf{z},\mathbf{y}_2} f^h(\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{y}_2)}{\sum_{\mathbf{z},\mathbf{y}_2} \prod_{i=1}^{n} f^h(\mathbf{z}_i, \mathbf{y}_2)} \tag{5}$$

with $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$. This scenario is explained in Fig. 3. Once a decomposition is performed, the algorithm is recursively applied to each and every potential obtained successively, until no further decomposition can be computed.

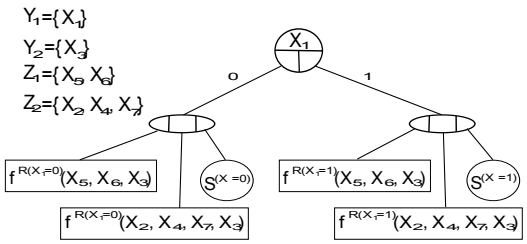


Figure 3: Example where a division of the graph has been found.

3.1 Computing the threshold ε

The value of ε should not be kept constant throughout the entire algorithm, as the range of possible values of the mutual information varies depending on the variables over which it is computed, and also on the computation of the mutual information conditioned to other variables. Thus, we propose assigning a rate δ, with $0 \leq \delta \leq 1$, and then compute ε as the fraction of the maximum mutual information determined by δ. In the case of unconditional mutual information between two variables X and Y, notice that

$$\begin{aligned} I(X,Y) &= H(X) - H(X|Y) \\ &= H(X) - (H(X,Y) - H(Y)) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned} \tag{6}$$

where $H(\cdot)$ denotes Shannon's entropy. Therefore, as $H(X) \leq \log |X|$, where $|X|$ is the number of possible values of X, it follows that

$I(X,Y) \leq \log |X| + \log |Y|$,

and thus, we compute ε as

$$\varepsilon := \delta \times (\log |X| + \log |Y|). \tag{7}$$

In the case of conditional mutual information, if we have X, Y and $\mathbf{Z}$, it follows that $I(X,Y|\mathbf{Z}) = H(X|\mathbf{Z}) - H(X,Y,\mathbf{Z})$. Since $H(X|\mathbf{Z}) = H(X,\mathbf{Z}) - H(\mathbf{Z})$ and $H(X|Y,\mathbf{Z}) = H(X,Y,\mathbf{Z}) - H(Y,\mathbf{Z})$, it holds that

$$\begin{aligned} I(X,Y|\mathbf{Z}) = H(X,\mathbf{Z}) - H(\mathbf{Z}) - H(X,Y,\mathbf{Z}) + \\ H(Y,\mathbf{Z}) \leq \log |X \cup \mathbf{Z}| + \log |Y \cup \mathbf{Z}| \end{aligned} \tag{8}$$

So, ε can be computed as:

$$\varepsilon := \delta \times (\log |X \cup \mathbf{Z}| + \log |Y \cup \mathbf{Z}|). \tag{9}$$

3.2 Detecting context specific independencies

Besides factorizations, RPTs can efficiently represent context specific independencies, in a similar way as they are represented by probability trees (Boutilier et al., 1996). A possible improvement of the described algorithm would be to widen the search by choosing at some point a variable Y_j connected to the rest of the variables and splitting the tree by it. In this case, the tree would grow with a Split node for such variable, and the process would continue for every branch, but now with the distributions restricted to the branch configuration. This way, the algorithm may follow different paths during factorization for each branch. In (Salmerón et al., 2000), a methodology of variable selection for labeling the internal nodes of probability trees is proposed. This methodology involves the calculation of the information gain of a given variable Y_j according to Eq. 9. Here we propose a similar approach to discern if splitting by a certain variable increases the quality of the learned model.

Let's assume that we are working with a potential f defined for a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$. Let's consider any variable $Y \in \mathbf{X}$ and define $\mathbf{Z} = \mathbf{X} \setminus \{Y\}$. The information gain resulting of splitting potential f by variable Y is computed as

$$\begin{aligned} I(Y,f) = S \cdot (\log |Y| - \log S) + \\ \sum_y S(Y=y) \log S(Y=y). \end{aligned} \tag{10}$$

The maximum possible value for $I(Y,f)$ can be obtained using the properties of Shannon's entropy. We define $N_Y = \sum_y S(Y=y)$. Then, it holds that

$$\frac{-1}{N_Y} \sum_{y \in \Omega_Y} S(Y=y) \log S(Y=y) \geq 0,$$

and, therefore

$$-\sum_{y\in\Omega_Y} S(Y=y)\log S(Y=y) \geq 0 \Rightarrow \sum_{y\in\Omega_Y} S(Y=y)\log S(Y=y) \leq 0. \quad (11)$$

Hence, replacing (11) in (10), we obtain $I(Y,f) \leq S\cdot(\log|Y| - \log S)$. Using this result, the criterion for choosing a variable to split from a parameter $0 \leq \delta \leq 1$ would be to select a variable Y if $I(Y,f) \geq \delta\cdot S\cdot(\log|Y| - \log S)$.

3.3 The algorithm

This section describes the pseudocode of the algorithm outlined above. The algorithm has been decomposed into different modules for the sake of readability. Algorithm 1 is the main body of the procedure, and the others are divided according to the different scenarios that can appear during the learning process. Algorithm 2 describes the required steps if the graph representing a potential is disconnected. In addition, Algorithm 3, gives the details of the process followed by a connected graph. Algorithm 3 is divided into several parts: Algorithm 4 shows what to do in case an Information Gain is detected for a variable. Algorithms 5 and 6 cover the case in which the resulting graph is disconnected within Algorithm 3.

```
Input: A potential f
Output: An RPT, recTree
begin
    Let G_f be the graph built for f
    if G_f is connected then
        if there are more than two variables in G_f then
            recTree = dealWithConnectedGraph()
        else
            Set recTree = PotentialTreeNode(f)
        end
    else
        recTree = dealWithDisconnectedGraph()
    end
end
```

Algorithm 1: Body of potential factorization algorithm

Example 1. We shall illustrate the algorithm with a simple example consisting of decomposing a potential f defined over four binary variables X, Y, Z, and W with values $f(x,y,z,w) = \{0.03,\ 0.04,\ 0.07,\ 0.06,\ 0.18,\ 0.24,\ 0.42,\ 0.36,\ 0.27,\ 0.36,\ 0.63,\ 0.54,\ 0.12,\ 0.16,\ 0.27,\ 0.24\}$. It can be seen that potential f is decomposable as $f(x,y,z,w) = f_1(x,y)f_2(z,w)$ with $f(x,y) = \{0.1,\ 0.6,\ 0.9,\ 0.4\}$ and $f_2(z,w) = \{0.3,\ 0.4,\ 0.7,\ 0.6\}$. Alg. 1 is called with potential f as argument. The first step is the construction of graph G_f. This is achieved by computing the entropy between each pair of variables and inserting the links for which the mutual information is greater than a given threshold. Let's assume we consider a threshold $\varepsilon = 1E-6$, that is, approximately equal to 0. We do this because independent variables can have a slightly positive mutual information due to rounding errors. The mutual information between each pair of variables, computed according to Eq. (1) is $I(X,Y) = 0.1484$, $I(X,Z) =$ 5.607E-17, $I(X,W) =$ 3.886E-17, $I(Y,Z) =$ 7.772E-17, $I(Y,W) = 0$, and $I(W,Z) = 0.0055$. Therefore, graph G_f has only two arcs, $X-Y$ and $W-Z$. Since the graph is disconnected, Alg.2 is called. As the two connected components of G_f only have two variables each, a list node is returned, in which the first factor is the marginal of f over (X,Y), the second is the marginal of f over (Z,W), and the third is a normalizing constant. More precisely, the three factors are: $g_1(x,y) = \{0.2,\ 1.2,\ 1.8,\ 0.8\}$, $g_2(z,w) = \{0.6,\ 0.8,\ 1.4,\ 1.2\}$, and $g_3(x,y,z,w) = 0.25$. It can be proved that $f(x,y,z,w) = g_1(x,y)g_2(z,w)g_3(x,y,z,w)$.

```
Input: A list C, which is the list of connected components
       of G_f, the potential f
Output: A List Tree Node, L
begin
    Let L be a List Tree Node;
    for each C_i in C do
        Let X_C be the variables in C_i;
        if C_i contains only 1 or 2 variables then
            Set recTree = f^{↓X_{C_i}};
        else
            recTree = PotentialFactorization(f^{↓X_{C_i}})
            (Alg.1);
        end
    end
    Add recTree to L;
    Let factor be a ValueTreeNode, computed as in
    Eq. (3);
    Add factor to L;
end
```

Algorithm 2: DealWithDisconnectedGraph()

Example 2. Now consider a potential f defined for three binary variables X, Y, and Z, with values $f(x,y,z) = \{0.3,\ 0.3,\ 0.3,\ 0.3,\ 0.1,\ 0.2,\ 0.75,\ 0.25\}$. Let's assume we consider a

threshold $\varepsilon = 0.001$. Then, Alg. 1 generates a complete graph G_f, as the mutual information values are $I(X,Y) = 0.0398$, $I(X,Z) = 0.0122$, and $I(Y,Z) = 0.0212$. Next, Alg. 3 is called with G_f as an argument. This algorithm selects a variable according to the connectivity values defined in Eq.(4). These values are $V(X) = 0.052$, $V(Y) = 0.061$, and $V(Z) = 0.0334$. Therefore, the chosen variable is Y. Since Y is connected to the rest of the variables, it is inserted into $\mathbf{Y}_1$ and G_f^Y, consisting of a graph with variables X and Z, and a link between them, is generated. In the next step, the information gain is computed according to Eq.(10) and, since it is equal to 0.0993, Alg.4 is called. This last procedure constructs a split node with variable Y and two children, one for each possible value of Y. The two children are $g_1(x,z) = f^{R(Y=0)}$ and $g_2(x,z) = f^{R(Y=1)}$. Their values are $g_1(x,z) = \{0.3,\ 0.3,\ 0.1,\ 0.2\}$ and $g_2(x,z) = \{0.3,\ 0.3,\ 0.75,\ 0.25\}$.

```
Input: The potential f, graph G
Output: A Tree Node, recTree
begin
    Let Y1 and Y2 be empty vectors of variables;
    while G remains connected and f has more than 2
    variables and there is no information gain do
        Choose variable to remove, Y (Eq.4);
        if Y was connected to all other variables in G_f
        then
            Add Y to Y1;
        else
            Add Y to Y2;
        end
        Build G_f^Y;
        if G_f^Y is connected then
            if There is Information Gain then
                treeNode =
                dealWithIndependentSplit()
            end
        else
            if Y1 is empty then
                treeNode = dealWithoutSplitChain
            else
                treeNode = dealWithSplitChain
            end
        end
    end
end
```

Algorithm 3: DealWithConnectedGraph()

```
Input: The potential f, Vector Y1
Output: A Split Tree Node, recTree
begin
    Let recTree be a Split Chain of variables from Y1;
    for each possible value y1 of Y1 do
        treeNode = potentialFactorization(f^{R(Y1=y1)});
        Update recTree's current leaf with f^{R(Y1=y1)};
    end
end
```

Algorithm 4: DealWithIndependentSplit()

```
Input: A list C which is the list of connected components
       of G_f, the potential f
Output: A Tree Node, recTree
begin
    if C has more than one element then
        Let recTree be a List Tree Node;
        for each C_i of C do
            Set f_1 = potentialFactorization(f^{↓X_{C_i}});
            Add f_1 as children of recTree;
        end
        Let factor be a ValueTreeNode, computed as in
        Eq. (3);
        Add factor to recTree;
    else
        Set recTree = PotentialTreeNode(f);
    end
end
```

Algorithm 5: DealWithoutSplitChain()

```
Input: The potential f, Vector Y1, Vector Y2, A list C
       which is the list of connected components of G_f
Output: A Tree Node, recTree
begin
    Let recTree be a Split Chain of variables from Y1;
    for each possible value y1 of Y1 do
        Let f_1 be f^{R(Y1=y1)};
        if C has more than one element then
            Let recTree be a List Tree Node;
            for each element C_i of C do
                Set f^R = f^{↓X_{C_i ∪ Y2}};
                Set f_1 = potentialFactorization(f^R);
                Add f_1 as children of recTree;
            end
            Let factor be a ValueTreeNode, computed
            as in Eq. (3);
        else
            Set treeNode = potentialFactorization(f_1)
        end
        Update recTree's current leaf with treeNode;
    end
end
```

Algorithm 6: DealWithSplitChain()

4 Experiments

4.1 Learning from a probability table

The first experiment consisted of 30 runs of the algorithm over randomly generated probability tables, defined over 6 binary variables. For each run, ϵ was set between 0 and 0.01 at intervals of 0.001, and the root mean squared error (MSE) between the original probability table and the learned RPT was computed. The results are shown in Fig. 4, where it can be seen that for higher values of ϵ, the approximations obtained are worse. But there is a point where the error suddenly increases, fact that can be used as a stopping criterion when searching for a solu-

tion. It was also observed that for higher values of ϵ, the decompositions obtained are more factorized and we get close to a model with a list node containing one factor per variable, which is not accurate and gives high error rates.

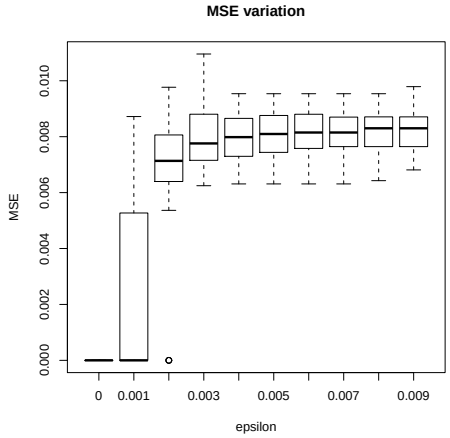


Figure 4: Results for the first experiment

4.2 Capturing repeated values

The second experiment consisted of 30 runs of the algorithm over probability tables for 6 binary variables generated from pruned probability trees, in the first case with a prune value (Salmerón et al., 2000) of 0.001 (light pruning) and in the second case, of 0.01 (severe pruning). Pruning the tree consists of replacing subtrees with the average value in the leaves, which means that the value for every configuration of variables in the pruned sub-tree becomes constant. Therefore, if we construct a table from such a tree, as a result all the cells in the table corresponding to the configurations in the pruned sub-tree will contain the same value. Thus, a severe pruning will generate more repeated values in the equivalent probability table than a light pruning. For each probability table generated, the algorithm is applied ten times, corresponding to ϵ values ranging from 0.0 to 0.01 at intervals of 0.001. Fig. 5 shows the MSE variation for each case. The upper panel of Fig. 5, which corresponds to light prune, shows that higher error values are reached in this case. So, it seems likely that the algorithm is able to detect this kind of regularity within a potential.

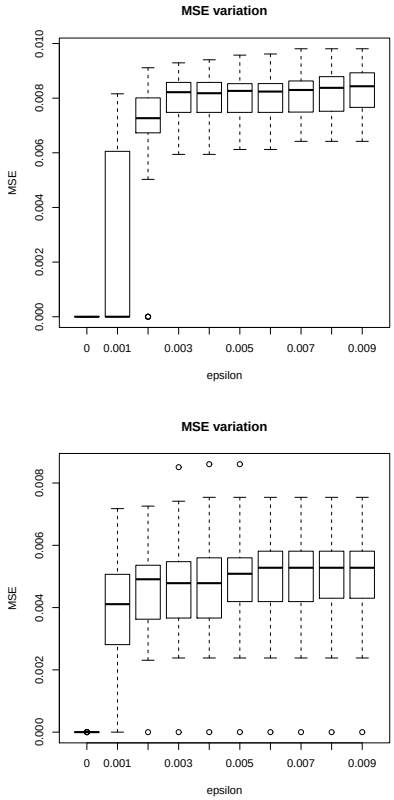


Figure 5: MSE variation learning from the same tree with light and severe prune

Other experiments have been performed to check the relation between accuracy and size of the learned model. The results show that the less accurate representations are those with smaller size. This reflects mainly the fact that smaller representations are expected to be a list of small factors (most likely, one per variable in the distribution), which is a not very accurate representation.

4.3 Learning from the same model

Due to the nature of the algorithm and the RPTs themselves, it is possible to obtain different RPTs representing the same distribution. The aim of this experiment was to check if those representations, while different, were accurate. A probability table was generated from a pruned probability tree. The algorithm was applied to it, with an ϵ value of 0.002, in order to get a slightly different distribution. The resulting RPT was called RPT_1. The algorithm

used the probability table represented by RPT_1 as an argument, this time with an ϵ value of 0, and returned a new RPT called RPT_2. This procedure was repeated 30 times, and then we calculated the mean and the standard deviation of both the difference in sizes of the resultant trees, and the Kullback-Leibler divergence relative to the original distribution. For the tree size differences, we got a mean of 0.73333, and a standard deviation of 1.311312. For the KL divergence, the mean downs to 0.021955 and the standard deviation to 0.040008. These results seem to confirm that under these circumstances, RPT_1 and RPT_2 are similar representations of the same distribution. In other words, this experiment illustrates the ability of the algorithm to find a RPT representation of a probability distribution, close to the original distribution.

5 Conclusions

In this paper we have proposed an algorithm for transforming a probabilistic potential into a RPT. The experiments performed suggest that the proposed algorithm is able to capture most of the details of the original distribution. This proposal can be used as the basis for designing approximate algorithms for inference in Bayesian networks, using RPT-based representations of the potentials involved in the inference process. The applicability of this method is limited, in practice, by the size of the potential that is going to be transformed into a RPT. It can be seen in Eq. (1), where the distributions used are obtained by marginalizing the original potential f. Therefore, the availability of a representation of f that allows an efficient computation of marginals would benefit the performance of the algorithm.

We are currently considering the possibility of extending the algorithm in order to learn directly from a database. Also, we are studying a way to detect a different kind of regularity, namely, the proportionality between different parts of the potential.

Acknowledgments

This research was jointly supported by the Spanish Ministry of Education and Science under projects TIN2007-67418-C03-03,02, the European Regional Development Fund (FEDER), the FPI scholarship programme (BES-2008-002049) and the Andalusian Research Program under project P08-TIC-03717.

References

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. 1996. Context-specific independence in Bayesian networks. In E. Horvitz and F.V. Jensen, editors, *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 115–123. Morgan & Kaufmann.

A. Cano, S. Moral, and A. Salmerón. 2000. Penniless propagation in join trees. *International Journal of Intelligent Systems*, 15:1027–1059.

A. Cano, M. Gómez-Olmedo, S. Moral, and C.B. Pérez-Ariza. 2009. Recursive probability trees for Bayesian networks. *CAEPIA 2009. Lecture Notes in Artificial Intelligence.*, 5988:242–251.

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

I. Martínez, S. Moral, C. Rodríguez, and A. Salmerón. 2002. Factorisation of probability trees and its application to inference in Bayesian networks. In J.A. Gámez and A. Salmerón, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models*, pages 127–134.

I. Martínez, S. Moral, C. Rodríguez, and A. Salmerón. 2005. Approximate factorisation of probability trees. *ECSQARU'05. Lecture Notes in Artificial Intelligence*, 3571:51–62.

I. Martínez, C. Rodríguez, and A. Salmerón. 2006. Dynamic importance sampling in Bayesian networks using factorisation of probability trees. In *Proceedings of the Third European Workshop on Probabilistic Graphical Models (PGM'06)*, pages 187–194.

J.R. Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1:81–106.

A. Salmerón, A. Cano, and S. Moral. 2000. Importance sampling in Bayesian networks using probability trees. *Computational Statistics and Data Analysis*, 34:387–413.

J.Q. Smith and P.E. Anderson. 2008. Conditional independence and chain event graphs. *Artificial Intelligence*, 172:42–68.

Likelihood-based inference for probabilistic graphical models: Some preliminary results

Marco E. G. V. Cattaneo
Department of Statistics, LMU Munich, Germany
cattaneo@stat.uni-muenchen.de

Abstract

A method for calculating some profile likelihood inferences in probabilistic graphical models is presented and applied to the problem of classification. It can also be interpreted as a method for obtaining inferences from hierarchical networks, a kind of imprecise probabilistic graphical models.

1 Introduction

The main result of the present paper is a method for calculating profile likelihood functions for an important class of probabilistic inferences, when the probabilities of a Bayesian network are learned from data. This result can also be interpreted as a method for obtaining inferences from hierarchical networks, which are networks with imprecisely known probabilities.

Likelihood-based inference is briefly outlined in the next section, while Section 3 contains the main result, stated in Theorem 1 (whose proof will be given in an extended version of the paper). Section 4 presents an application of this result to the problem of classification.

2 Likelihood

Let P_θ be a parametric probabilistic model for some discrete random variables $X, Y, \ldots$, where $\theta \in \Theta$ is the parameter (vector) and Θ is the parameter space. The observation of a realization $X = x$ induces the (normalized) likelihood function lik on Θ defined by

$$lik(\theta) = \frac{P_\theta(X = x)}{\sup_{\theta' \in \Theta} P_{\theta'}(X = x)}$$

for all $\theta \in \Theta$. Moreover, when $X = x$ is observed, the model P_θ is updated into the conditional model $P_\theta(\,\cdot \,|\, X = x)$.

The inference about the value $g(\theta)$ of a function $g : \Theta \to \mathcal{G}$ (where $\mathcal{G}$ can be any set) can be based on the (normalized) profile likelihood function lik_g on $\mathcal{G}$ defined by

$$lik_g(\gamma) = \sup_{\theta \in \Theta \,:\, g(\theta) = \gamma} lik(\theta)$$

for all $\gamma \in \mathcal{G}$, where $\sup \varnothing$ is interpreted as 0. In particular, if there is a unique $\gamma \in \mathcal{G}$ such that $lik_g(\gamma) = 1$, then γ is the maximum likelihood estimate of $g(\theta)$. More generally, the likelihood-based confidence region for $g(\theta)$ with cutoff point $\beta \in [0, 1[$ is the set

$$\{\gamma \in \mathcal{G} : lik_g(\gamma) > \beta\},$$

whose confidence level can often be approximated thanks to the result of (Wilks, 1938). These are the usual likelihood-based point estimates and set estimates; more general ways of basing inferences and decisions directly on the likelihood function are discussed in (Cattaneo, 2007).

Example 1. Let $X_1, \ldots, X_{10}$ be 10 categorical variables taking values in the set $\{a, b, c\}$. The assumption that $X_1, \ldots, X_{10}$ are independent and identically distributed leads to a parametric probabilistic model P_θ with $P_\theta(X_i = \omega) = \theta_\omega$, where the parameter $\theta = (\theta_a, \theta_b, \theta_c)$ is an element of the standard 2-dimensional simplex

$$\Theta = \left\{(\theta_a, \theta_b, \theta_c) \in [0, 1]^3 : \theta_a + \theta_b + \theta_c = 1)\right\}.$$

Assume that the realizations of $X_1, \ldots, X_9$ are observed, and the values a, b, and c appear 2, 4, and 3 times, respectively. This observation

Pp. 57–65 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

induces the (normalized) multinomial likelihood function lik on Θ defined by

$$lik(\theta) = \tfrac{14348907}{1024}\,\theta_a^2\,\theta_b^4\,\theta_c^3$$

for all $\theta \in \Theta$. The inference about the probability that the realization of X_{10} will be either a or b can be based on the profile likelihood function lik_g on $[0,1]$, with $g : \Theta \to [0,1]$ defined by

$$g(\theta) = P_\theta\left(X_{10} \in \{a,b\}\right) = \theta_a + \theta_b$$

for all $\theta \in \Theta$ (note that conditioning P_θ on the observed realizations of $X_1, \ldots, X_9$ has no influence on the probability distribution of X_{10}). Since the (normalized) profile likelihood function lik_g on $[0,1]$ satisfies

$$lik_g(\gamma) = \tfrac{19683}{64}\,\gamma^6\,(1-\gamma)^3$$

for all $\gamma \in [0,1]$, the maximum likelihood estimate of $P_\theta\left(X_{10} \in \{a,b\}\right)$ is $\widehat{\gamma} = \frac{2}{3}$, while for instance $[0.35, 0.90]$ is an approximate 95% confidence interval for $P_\theta\left(X_{10} \in \{a,b\}\right)$, since it corresponds approximately to the likelihood-based confidence region with cutoff point $\beta = 0.15$.

2.1 Hierarchical model

The parametric probabilistic model and the likelihood function can be considered as the two levels of a hierarchical model, in which the likelihood function describes the relative plausibility of the parameter values. As noted above, when $X = x$ is observed, the set $\{P_\theta : \theta \in \Theta\}$ is updated by conditioning each element P_θ on the observed event: this corresponds to the updating of the imprecise Bayesian model studied in (Walley, 1991). Hence, the hierarchical model generalizes the imprecise Bayesian model, in the sense that the second level (that is, the likelihood function) describes additional information about the relative plausibility of the elements of $\{P_\theta : \theta \in \Theta\}$.

This additional information allows fundamental advantages of the hierarchical model over the imprecise Bayesian model, such as the possibility of starting without prior information and the increased robustness of the conclusions: see for example (Cattaneo, 2009). Moreover, since the membership functions of fuzzy sets are often interpreted as likelihood functions (the extension principle of possibility theory corresponds then to the use of profile likelihood functions), the hierarchical model can handle fuzzy data and possibilistic information or variables in a unified and well-founded way: see for instance (Cattaneo, 2008).

3 Networks

When X is a categorical variable, let Ω_X denote the set of all possible realizations of X. Moreover, let $\mathcal{F}_X$ denote the set of all possible real functions on Ω_X, and let $\mathcal{S}_X$ denote the set of all possible probability distributions on Ω_X. Hence, $\mathcal{S}_X \subset \mathcal{F}_X$, and $\mathcal{S}_X$ can be identified with the standard simplex of dimension $|\Omega_X| - 1$, where $|\Omega_X|$ denotes the cardinality of Ω_X. Moreover, let 0_X denote the function on Ω_X with constant value 0 (therefore, $0_X \in \mathcal{F}_X$, but $0_X \notin \mathcal{S}_X$). Finally, if $f \in \mathcal{F}_X$ with $f(x) \geqslant 0$ for all $x \in \Omega_X$, then let $\langle f \rangle$ denote the probability distribution on Ω_X proportional to f when $f \neq 0_X$, and the uniform probability distribution on Ω_X when $f = 0_X$. That is, $\langle f \rangle \in \mathcal{S}_X$, and for all $x \in \Omega_X$,

$$\langle f \rangle(x) = \begin{cases} \frac{f(x)}{\sum_{x' \in \Omega_X} f(x')} & \text{if } \sum_{x' \in \Omega_X} f(x') > 0, \\ \frac{1}{|\Omega_X|} & \text{if } \sum_{x' \in \Omega_X} f(x') = 0. \end{cases}$$

Let $X_1, \ldots, X_k$ be k categorical variables such that $|\Omega_{X_i}| \geqslant 2$ for all $i \in \{1, \ldots, k\}$. Assumptions about conditional independencies among the variables $X_1, \ldots, X_k$ can be encoded in a directed acyclic graph G with nodes $X_1, \ldots, X_k$: see for example (Jensen and Nielsen, 2007). Let Π_i denote the joint variable composed of all parents of X_i according to G, where Π_i is assumed to be constant when X_i has no parents. The other component of a Bayesian network, besides the graph G, are the probability distributions of X_i conditional on $\Pi_i = \pi_i$, for each $i \in \{1, \ldots, k\}$ and each $\pi_i \in \Omega_{\Pi_i}$. Altogether, these conditional probability distributions can be described by the parameter $\theta \in \Theta_G$, where

$$\Theta_G = \bigtimes_{i=1}^{k} \bigtimes_{\pi_i \in \Omega_{\Pi_i}} \mathcal{S}_{X_i}$$

is a Cartesian product of the simplexes $\mathcal{S}_{X_i}$. For each $\theta \in \Theta_G$, let $\theta_{X_i|\pi_i}$ denote the corresponding probability distribution of X_i conditional on $\Pi_i = \pi_i$ (hence, $\theta_{X_i|\pi_i} \in \mathcal{S}_{X_i}$). The Bayesian network described by the graph G and the parameter $\theta \in \Theta_G$ determines the joint probability distribution P_θ on $\Omega_{X_1} \times \cdots \times \Omega_{X_k}$ defined by

$$P_\theta(x_1, \ldots, x_k) = \prod_{i=1}^{k} \theta_{X_i|\pi_i}(x_i)$$

for all $(x_1, \ldots, x_k) \in \Omega_{X_1} \times \cdots \times \Omega_{X_k}$, where π_i are the corresponding realizations of the joint variables Π_i.

To allow uncertainty about the involved probability values, Bayesian networks have been generalized to credal networks, which can be described by a directed acyclic graph G and a set $\Theta \subseteq \Theta_G$ of parameters: see for instance (Antonucci and Zaffalon, 2008). A credal network determines an imprecise Bayesian model $\{P_\theta : \theta \in \Theta\}$, instead of a single probability distribution P_θ. That is, a credal network corresponds mathematically to a set of Bayesian networks with the same graph G. A credal network is said to be separately specified if

$$\Theta = \bigtimes_{i=1}^{k} \bigtimes_{\pi_i \in \Omega_{\Pi_i}} \Theta_{X_i|\pi_i} \tag{1}$$

is the Cartesian product of the sets $\Theta_{X_i|\pi_i}$, with $\Theta_{X_i|\pi_i} \subseteq \mathcal{S}_{X_i}$ for all $i \in \{1, \ldots, k\}$ and all $\pi_i \in \Omega_{\Pi_i}$. That is, a separately specified credal network consists of all Bayesian networks with graph G and probability distributions of X_i conditional on $\Pi_i = \pi_i$ freely selected from the sets $\Theta_{X_i|\pi_i}$ (note that only the so-called strong extension of a separately specified credal network is considered in the present paper).

To allow additional information about the relative plausibility of the involved probability values, credal networks have been generalized to hierarchical networks, which can be described by a directed acyclic graph G, a set $\Theta \subseteq \Theta_G$ of parameters, and a (normalized) likelihood function lik on Θ: see for example (Cattaneo, 2009). A hierarchical network determines a hierarchical model with as first level the imprecise Bayesian model $\{P_\theta : \theta \in \Theta\}$, and as second level the likelihood function lik on Θ, describing the relative plausibility of the elements of $\{P_\theta : \theta \in \Theta\}$. That is, a hierarchical network corresponds mathematically to a set of Bayesian networks with the same graph G but in general with different degrees of plausibility. A hierarchical network is said to be separately specified if Θ satisfies (1), and lik is the product of the local likelihood functions $lik_{X_i|\pi_i}$ on $\Theta_{X_i|\pi_i}$, in the sense that

$$lik(\theta) = \prod_{i=1}^{k} \prod_{\pi_i \in \Omega_{\Pi_i}} lik_{X_i|\pi_i}(\theta_{X_i|\pi_i})$$

for all $\theta \in \Theta$, with $\Theta_{X_i|\pi_i} \subseteq \mathcal{S}_{X_i}$ and $lik_{X_i|\pi_i} : \Theta_{X_i|\pi_i} \to [0, 1]$ for all $i \in \{1, \ldots, k\}$ and all $\pi_i \in \Omega_{\Pi_i}$. That is, the separately specified hierarchical networks generalize the separately specified credal networks by adding information about the relative plausibility of the elements of the sets $\Theta_{X_i|\pi_i}$.

3.1 Learning probabilities from data

Learning networks from data is a fundamental problem. In the present paper, only the simplest case is considered: the directed acyclic graph G is assumed known, and the dataset is complete. That is, the dataset consists of n realizations of the joint variable $X = (X_1, \ldots, X_k)$. For each $i \in \{1, \ldots, k\}$ and each $\pi_i \in \Omega_{\Pi_i}$, let $n_{X_i|\pi_i}$ denote the function on Ω_{X_i} assigning to each $x_i \in \Omega_{X_i}$ the number of realizations of the joint variable X such that $X_i = x_i$ and $\Pi_i = \pi_i$. Hence, $n_{X_i|\pi_i} \in \mathcal{F}_{X_i}$, and for all $i \in \{1, \ldots, k\}$,

$$\sum_{\pi_i \in \Omega_{\Pi_i}} \sum_{x_i \in \Omega_{X_i}} n_{X_i|\pi_i}(x_i) = n. \tag{2}$$

When the n realizations of the joint variable X are considered independent and identically distributed according to the joint probability distribution P_θ with $\theta \in \Theta_G$, they induce the (normalized) likelihood function lik on Θ_G defined by

$$lik(\theta) = \prod_{i=1}^{k} \prod_{\pi_i \in \Omega_{\Pi_i}} \prod_{x_i \in \Omega_{X_i}} \frac{\big(\theta_{X_i|\pi_i}(x_i)\big)^{n_{X_i|\pi_i}(x_i)}}{\big(\langle n_{X_i|\pi_i} \rangle(x_i)\big)^{n_{X_i|\pi_i}(x_i)}}$$

for all $\theta \in \Theta_G$, where 0^0 is interpreted as 1. The denominators of the fractions normalize the likelihood function, since lik is maximized by the parameter $\widehat{\theta} \in \Theta_G$ such that $\widehat{\theta}_{X_i|\pi_i} = \langle n_{X_i|\pi_i} \rangle$ for all $i \in \{1, \ldots, k\}$ and all $\pi_i \in \Omega_{\Pi_i}$. However, $\widehat{\theta}$ is the unique parameter maximizing lik only if $n_{X_i|\pi_i} \neq 0_{X_i}$ for all $i \in \{1, \ldots, k\}$ and all $\pi_i \in \Omega_{\Pi_i}$ (that is, only if all possible realizations $\Pi_i = \pi_i$ appear at least once in the dataset).

Hence, the likelihood function lik on Θ_G factorizes in the local likelihood functions $lik_{X_i|\pi_i}$ on $\mathcal{S}_{X_i}$ defined by

$$lik_{X_i|\pi_i}(\theta_{X_i|\pi_i}) = \prod_{x_i \in \Omega_{X_i}} \frac{\big(\theta_{X_i|\pi_i}(x_i)\big)^{n_{X_i|\pi_i}(x_i)}}{\big(\langle n_{X_i|\pi_i} \rangle(x_i)\big)^{n_{X_i|\pi_i}(x_i)}}$$

for all $\theta_{X_i|\pi_i} \in \mathcal{S}_{X_i}$. Estimates of the conditional probability distributions $\theta_{X_i|\pi_i}$ can easily be based on the multinomial likelihood functions $lik_{X_i|\pi_i}$: if $n_{X_i|\pi_i} \neq 0$, then $\widehat{\theta}_{X_i|\pi_i} = \langle n_{X_i|\pi_i} \rangle$ is the maximum likelihood estimate; alternatively, $lik_{X_i|\pi_i}$ can be combined with a prior probability distribution on $\mathcal{S}_{X_i}$ (usually a Dirichlet distribution) to obtain a Bayesian estimate of $\theta_{X_i|\pi_i}$. However, the Bayesian network corresponding to the estimated conditional probability distributions does not contain any information about the uncertainty of those estimates, and consequently it does not contain any information about the uncertainty of the resulting probabilistic inferences.

To include some information about the uncertainty of the conditional probability distributions and of the resulting probabilistic inferences, set estimates $\Theta_{X_i|\pi_i} \subseteq \mathcal{S}_{X_i}$ of the conditional probability distributions $\theta_{X_i|\pi_i}$ can be based on the local likelihood functions $lik_{X_i|\pi_i}$ (instead of point estimates $\theta_{X_i|\pi_i} \in \mathcal{S}_{X_i}$). The set estimates $\Theta_{X_i|\pi_i}$ determine a separately specified credal network, and inferences can then be based on the corresponding imprecise Bayesian model. The usual way of obtaining an imprecise probability distribution $\Theta_{X_i|\pi_i}$ from a multinomial likelihood function $lik_{X_i|\pi_i}$ is by combining it with a set of prior Dirichlet distributions, called imprecise Dirichlet model: see for example (Walley, 1996). But the confidence level of the set estimates $\Theta_{X_i|\pi_i}$ obtained from the imprecise Dirichlet model can be arbitrarily low (for sufficiently large n: compare with Example 2): see for instance Wilson's comment in the discussion of (Walley, 1996). To avoid this problem, the sets $\Theta_{X_i|\pi_i} \subseteq \mathcal{S}_{X_i}$ could be estimated as likelihood-based confidence regions for $\theta_{X_i|\pi_i}$, according to the multinomial likelihood functions $lik_{X_i|\pi_i}$, but in general the closure of the resulting set estimates $\Theta_{X_i|\pi_i}$ would be convex with infinitely many extreme points when $|\Omega_{X_i}| \geqslant 3$, and this would lead to computational difficulties.

Instead of reducing them to likelihood-based confidence regions $\Theta_{X_i|\pi_i}$, it is better to maintain the whole likelihood functions $lik_{X_i|\pi_i}$ as descriptions of the uncertainty about the conditional probability distributions $\theta_{X_i|\pi_i}$. The likelihood function lik on Θ_G describes then the uncertainty about the whole Bayesian network (given the graph G), and corresponds to a separately specified hierarchical network with $\Theta = \Theta_G$. Learning hierarchical networks from data is straightforward (when the graph G is assumed known), and no estimates or prior distributions are necessary, but in general the calculation of profile likelihood functions (on which inferences and decisions are based) is not so simple. However, the following theorem shows that for particular classes of functions g on Θ, obtaining the profile likelihood function lik_g is straightforward too.

Theorem 1. *For each $i \in \{1, \ldots, k\}$ and each $\pi_i \in \Omega_{\Pi_i}$, let $d_{X_i|\pi_i}, q_{X_i|\pi_i} \in \mathcal{F}_{X_i}$ with $d_{X_i|\pi_i}(x_i) \geqslant 0$ for all $x_i \in \Omega_{X_i}$. Moreover, let $lik : \Theta_G \to [0,1]$ and $g : \Theta_G \to [0,+\infty]$ be defined by*

$$lik(\theta) = \prod_{i=1}^{k} \prod_{\pi_i \in \Omega_{\Pi_i}} \prod_{x_i \in \Omega_{X_i}} \frac{\big(\theta_{X_i|\pi_i}(x_i)\big)^{d_{X_i|\pi_i}(x_i)}}{\big(\langle d_{X_i|\pi_i} \rangle(x_i)\big)^{d_{X_i|\pi_i}(x_i)}}$$

and

$$g(\theta) = \prod_{i=1}^{k} \prod_{\pi_i \in \Omega_{\Pi_i}} \prod_{x_i \in \Omega_{X_i}} \big(\theta_{X_i|\pi_i}(x_i)\big)^{q_{X_i|\pi_i}(x_i)},$$

respectively, for all $\theta \in \Theta_G$, where 0^0 is interpreted as 1, and 0^x is interpreted as $+\infty$ for all

negative x (but $g(\theta)$ is undefined when both 0 and $+\infty$ appear in the same product). Finally, let $\underline{\alpha}$ and $\overline{\alpha}$ be the infimum and the supremum, respectively, of the set

$$\left\{\alpha \in \mathbb{R} : (d_{X_i|\pi_i} + \alpha\, q_{X_i|\pi_i})(x_i) \geqslant 0 \ \forall\, i, \pi_i, x_i\right\}$$

(linear combinations of functions are to be interpreted pointwise).

- *If $\underline{\alpha} = \overline{\alpha} = 0$, then $lik_g(\gamma) = 1$ for all $\gamma \in [0, +\infty]$.*
- *Otherwise, define $\underline{\theta}, \theta[\alpha], \overline{\theta} \in \Theta_G$ as follows:*

$$\underline{\theta}_{X_i|\pi_i} = \begin{cases} \langle d_{X_i|\pi_i} + \underline{\alpha}\, q_{X_i|\pi_i} \rangle & \text{if } \underline{\alpha} \neq -\infty \\ \quad \text{and } d_{X_i|\pi_i} + \underline{\alpha}\, q_{X_i|\pi_i} \neq 0_{X_i}, & \\ \langle q_{X_i|\pi_i} \rangle & \text{if } \underline{\alpha} \neq -\infty \\ \quad \text{and } d_{X_i|\pi_i} + \underline{\alpha}\, q_{X_i|\pi_i} = 0_{X_i}, & \\ \langle -q_{X_i|\pi_i} \rangle & \text{if } \underline{\alpha} = -\infty \\ & \text{and } q_{X_i|\pi_i} \neq 0_{X_i}, \\ \langle d_{X_i|\pi_i} \rangle & \text{if } \underline{\alpha} = -\infty \\ & \text{and } q_{X_i|\pi_i} = 0_{X_i}, \end{cases}$$

$$\theta[\alpha]_{X_i|\pi_i} = \langle d_{X_i|\pi_i} + \alpha\, q_{X_i|\pi_i} \rangle,$$

$$\overline{\theta}_{X_i|\pi_i} = \begin{cases} \langle d_{X_i|\pi_i} + \overline{\alpha}\, q_{X_i|\pi_i} \rangle & \text{if } \overline{\alpha} \neq +\infty \\ \quad \text{and } d_{X_i|\pi_i} + \overline{\alpha}\, q_{X_i|\pi_i} \neq 0_{X_i}, & \\ \langle -q_{X_i|\pi_i} \rangle & \text{if } \overline{\alpha} \neq +\infty \\ \quad \text{and } d_{X_i|\pi_i} + \overline{\alpha}\, q_{X_i|\pi_i} = 0_{X_i}, & \\ \langle q_{X_i|\pi_i} \rangle & \text{if } \overline{\alpha} = +\infty \\ & \text{and } q_{X_i|\pi_i} \neq 0_{X_i}, \\ \langle d_{X_i|\pi_i} \rangle & \text{if } \overline{\alpha} = +\infty \\ & \text{and } q_{X_i|\pi_i} = 0_{X_i}, \end{cases}$$

respectively, for all $\alpha \in]\underline{\alpha}, \overline{\alpha}[$, all $i \in \{1, \ldots, k\}$, and all $\pi_i \in \Omega_{\Pi_i}$.

Then $lik_g(g(\underline{\theta})) = lik(\underline{\theta})$ and $lik_g(g(\overline{\theta})) = lik(\overline{\theta})$.

If $g(\underline{\theta}) > 0$, then for all $\gamma \in [0, g(\underline{\theta})[$,

$$lik_g(\gamma) = \begin{cases} \left(\frac{\gamma}{g(\underline{\theta})}\right)^{-\underline{\alpha}} lik(\underline{\theta}) & \text{if } \underline{\alpha} \neq -\infty, \\ 0 & \text{if } \underline{\alpha} = -\infty. \end{cases}$$

If $g(\underline{\theta}) < g(\overline{\theta})$, then the graph of the restriction of lik_g to $]g(\underline{\theta}), g(\overline{\theta})[$ is the set

$$\{(g(\theta[\alpha]), lik(\theta[\alpha])) : \alpha \in]\underline{\alpha}, \overline{\alpha}[\,\},$$

and $g(\theta[\alpha])$ is a continuous, strictly increasing function of $\alpha \in]\underline{\alpha}, \overline{\alpha}[$.

If $g(\overline{\theta}) < +\infty$, then for all $\gamma \in]g(\overline{\theta}), +\infty]$,

$$lik_g(\gamma) = \begin{cases} \left(\frac{\gamma}{g(\overline{\theta})}\right)^{-\overline{\alpha}} lik(\overline{\theta}) & \text{if } \overline{\alpha} \neq +\infty, \\ 0 & \text{if } \overline{\alpha} = +\infty. \end{cases}$$

The likelihood function *lik* of Theorem 1 generalizes the likelihood function on Θ_G induced by a complete dataset, for which the functions $d_{X_i|\pi_i} = n_{X_i|\pi_i}$ can take only integer values and must satisfy conditions such as (2). The function g of Theorem 1 can for example describe the probability of a particular realization $(x_1, \ldots, x_k) \in \Omega_{X_1} \times \cdots \times \Omega_{X_k}$ of the joint variable $X = (X_1, \ldots, X_k)$; that is,

$$g(\theta) = P_\theta(X_1 = x_1, \ldots, X_k = x_k)$$

for all $\theta \in \Theta_G$. In this case, $q_{X_i|\pi_i} = n_{X_i|\pi_i}$ for the particular dataset consisting of the single realization $(x_1, \ldots, x_k)$ of the joint variable X. If the functions $n'_{X_i|\pi_i}$ describe a second dataset consisting of the single realization $(x'_1, x_2, \ldots, x_k)$ of the joint variable X, then the function g with $q_{X_i|\pi_i} = n_{X_i|\pi_i} - n'_{X_i|\pi_i}$ satisfies

$$g(\theta) = \frac{P_\theta(X_1 = x_1 \,|\, X_2 = x_2, \ldots, X_k = x_k)}{P_\theta(X_1 = x'_1 \,|\, X_2 = x_2, \ldots, X_k = x_k)}$$

for all $\theta \in \Theta_G$ such that the right-hand side is well-defined. That is, g describes the probability ratio of the possible realizations x_1 and x'_1 of X_1 conditional on the realizations of $X_2, \ldots, X_k$. In the next section, Theorem 1 with this kind of function g is used in the problem of classification: the goal is to determine the realization of X_1 given the realizations of $X_2, \ldots, X_k$.

The formulation of Theorem 1 is rather complex, because several special cases must be considered, but the central part of the theorem is pretty simple: it is the parametric expression for the graph of the profile likelihood function lik_g restricted to the interval $]g(\underline{\theta}), g(\overline{\theta})[$. For example, in the problem of classification studied in the next section, it suffices to consider this central part, since $]g(\underline{\theta}), g(\overline{\theta})[=]0, +\infty[$.

The idea behind the parametric expression of the graph of lik_g is the following: if $\theta[\alpha]$ maximizes $(g(\theta))^\alpha\ lik(\theta)$ over all $\theta \in \Theta_G$ for some $\alpha \in \mathbb{R}$, then $\theta[\alpha]$ maximizes $lik(\theta)$ over all $\theta \in \Theta_G$ such that $g(\theta) = g(\theta[\alpha])$, and therefore $lik_g(g(\theta[\alpha])) = lik(\theta[\alpha])$. For the particular classes of functions lik and g considered in Theorem 1, finding the parameter $\theta[\alpha]$ maximizing $(g(\theta))^\alpha\ lik(\theta)$ over all $\theta \in \Theta_G$ is extremely simple. For more general classes of functions lik and g, the above idea can be combined with approximation algorithms for maximizing $(g(\theta))^\alpha\ lik(\theta)$, such as the EM algorithm of (Dempster et al., 1977), but this goes beyond the scope of the present paper.

4 Naive classifiers

Let $C, F_1, \ldots, F_{k-1}$ be k categorical variables. The variables $F_1, \ldots, F_{k-1}$ describe $k-1$ features of an object, while C is the variable of interest: it describes the object's class. Having observed m features of an object, say $F_1 = f_1, \ldots, F_m = f_m$, with $m \in \{0, \ldots, k-1\}$, the goal is to classify it; that is, to predict the realization of C. The problem is particularly simple if the features $F_1, \ldots, F_{k-1}$ are assumed to be conditionally independent given the class C. This assumption can be encoded in the directed acyclic graph G_N with nodes $C, F_1, \ldots, F_{k-1}$ such that C has no parents and is the only parent of $F_1, \ldots, F_{k-1}$.

The Bayesian network described by the graph G_N and a parameter $\theta \in \Theta_{G_N}$ is called naive Bayes classifier (NBC): such classifiers were proposed in (Duda and Hart, 1973). For each pair of different classes $a, b \in \Omega_C$, let $g_{a,b} : \Theta_{G_N} \to [0, +\infty]$ be defined by

$$g_{a,b}(\theta) = \frac{P_\theta(C = a, F_1 = f_1, \ldots, F_m = f_m)}{P_\theta(C = b, F_1 = f_1, \ldots, F_m = f_m)}$$

for all $\theta \in \Theta_G$, where $\frac{x}{0}$ is interpreted as $+\infty$ for all positive x, and as 1 when $x = 0$. A strict partial preference order $\succ$ on Ω_C is obtained by considering the values $g_{a,b}(\theta)$, for the parameter θ of the Bayesian network and all pairs of different classes $a, b \in \Omega_C$: if $g_{a,b}(\theta) > 1$, then $a \succ b$ (that is, a is preferred to b), while if $g_{a,b}(\theta) < 1$, then $b \succ a$; finally, if $g_{a,b}(\theta) = 1$, then there is no preference between a and b. The NBC returns as prediction of C the maximal elements of Ω_C according to $\succ$ (that is, the $c \in \Omega_C$ such that there is no $c' \in \Omega_C$ with $c' \succ c$). Usually the prediction consists of a single class, but sometimes it can consist of several classes (with no preference among them). The parameter θ of the Bayesian network can be estimated from training data (for example by maximum likelihood estimation: see Subsection 3.1), but the resulting NBC does not contain any information about the uncertainty of the estimate θ and of the inferred values $g_{a,b}(\theta)$.

The credal network described by the graph G_N and a set $\Theta \subseteq \Theta_{G_N}$ of parameters is called naive credal classifier (NCC): such classifiers were proposed in (Zaffalon, 2002). A strict partial preference order $\succ$ on Ω_C (called credal dominance) is obtained by considering the values $g_{a,b}(\theta)$, for all parameters $\theta \in \Theta$ and all pairs of different classes $a, b \in \Omega_C$: there is a preference between a and b only if either $g_{a,b}(\theta) > 1$ for all $\theta \in \Theta$ (in which case $a \succ b$), or $g_{a,b}(\theta) < 1$ for all $\theta \in \Theta$ (in which case $b \succ a$). The NCC returns as prediction of C the maximal elements of Ω_C according to $\succ$; hence, the prediction often consists of more than one class. The set Θ of parameters can be estimated from training data (for example on the basis of the imprecise Dirichlet model: see Subsection 3.1): the resulting NCC contains some information about the uncertainty of the inferred values $g_{a,b}(\theta)$, and the number of classes returned as prediction of C depends on the amount of uncertainty (the more uncertainty, the more classes).

The hierarchical network described by the graph G_N and a (normalized) likelihood function lik on Θ_{G_N} can be called naive hierarchical classifier (NHC). For each $\beta \in [0, 1[$, a strict partial preference order $\succ_\beta$ on Ω_C is obtained by considering the profile likelihood functions $lik_{g_{a,b}}$ on $[0, +\infty]$, for all pairs of different classes $a, b \in \Omega_C$: there is a preference between a and b only if either $lik_{g_{a,b}}(\gamma) \leqslant \beta$ for all $\gamma \in [0, 1]$ (in which case $a \succ_\beta b$), or $lik_{g_{a,b}}(\gamma) \leqslant \beta$ for all $\gamma \in [1, +\infty]$ (in which case $b \succ_\beta a$). The NHC returns as prediction of C with cutoff point β

the maximal elements of Ω_C according to $\succ_\beta$. Hence, the prediction can consist of one or more classes, and the number of classes increases as β decreases, in the sense that additional classes can be included in the prediction as β decreases. The likelihood function lik on Θ_{G_N} can be induced by training data, and when lik satisfies the condition of Theorem 1, the profile likelihood functions $lik_{g_{a,b}}$ are easily obtained. In order to satisfy that condition, it is not necessary for the training dataset to be complete (the features of the objects in the dataset need not be observed), but when it is complete, Theorem 1 implies the following simple result.

Corollary 1. *Let $a, b \in \Omega_C$ be two different classes, and for both $c \in \{a, b\}$ and each $i \in \{1, \ldots, m\}$, let n_c and $n_{c,i}$ be the numbers of objects in the complete training dataset with $C = c$, and with $C = c$ and $F_i = f_i$, respectively. Moreover, define*

$$\underline{\alpha} = -\min\{n_a, n_{a,1}, \ldots, n_{a,m}\}$$

and

$$\overline{\alpha} = \min\{n_b, n_{b,1}, \ldots, n_{b,m}\}.$$

- *If $\underline{\alpha} = \overline{\alpha} = 0$, then $lik_{g_{a,b}}(\gamma) = 1$ for all $\gamma \in [0, +\infty]$.*
- *Otherwise, let $x_{a,b}, y_a, y_b : [\underline{\alpha}, \overline{\alpha}] \to [0, +\infty]$ be defined by*

$$x_{a,b}(\alpha) = \frac{n_a + \alpha}{n_b - \alpha} \prod_{i=1}^{m} \left(\frac{n_{a,i} + \alpha}{n_a + \alpha} \frac{n_b - \alpha}{n_{b,i} - \alpha} \right),$$

$$y_a(\alpha) = \frac{(n_a + \alpha)^{n_a}}{{n_a}^{n_a}} \prod_{i=1}^{m} \frac{{n_a}^{n_a} \, (n_{a,i} + \alpha)^{n_{a,i}}}{{n_{a,i}}^{n_{a,i}} \, (n_a + \alpha)^{n_a}},$$

$$y_b(\alpha) = \frac{(n_b - \alpha)^{n_b}}{{n_b}^{n_b}} \prod_{i=1}^{m} \frac{{n_b}^{n_b} \, (n_{b,i} - \alpha)^{n_{b,i}}}{{n_{b,i}}^{n_{b,i}} \, (n_b - \alpha)^{n_b}},$$

respectively, for all $\alpha \in [\underline{\alpha}, \overline{\alpha}]$, where 0^0 is interpreted as 1, and $\frac{x}{0}$ is interpreted as $+\infty$ for all positive x, and as 1 when $x = 0$.

Then $x_{a,b}$ is an increasing bijection, and the graph of $lik_{g_{a,b}}$ is the set

$$\{(x_{a,b}(\alpha),\, y_a(\alpha)\, y_b(\alpha)) : \alpha \in [\underline{\alpha}, \overline{\alpha}]\}.$$

If the NHC is learned from training data, then for sufficiently large $\beta \in [0, 1[$, the predictions with cutoff point β correspond to the ones returned by the NBC based on maximum likelihood estimation (if this is well-defined). But as β decreases, more and more classes are included in the predictions with cutoff point β; and for sufficiently small β, the predictions are vacuous, in the sense that they consist of all possible classes. Hence, the NHC learned from training data can be interpreted as a description of the uncertainty about the NBC based on maximum likelihood estimation: when the cutoff point $\beta \in]0, 1[$ is fixed, the numbers of classes in the predictions depend on the amount of uncertainty (the more uncertainty, the more classes). In particular, if c is the prediction of C returned by the NBC, then $\beta_c = \max_{c' \in C \setminus \{c\}} lik_{g_{c,c'}}(1)$ is the minimum value of $\beta \in]0, 1[$ such that the prediction of C with cutoff point β returned by the NHC is c as well. Therefore, β_c is an index of the uncertainty about the prediction c: the larger β_c, the more uncertainty; in fact, β_c is the likelihood ratio test statistic for the set of all parameters $\theta \in \Theta_{G_N}$ such that the corresponding NBC does not return c as prediction of C: see for instance (Wilks, 1938).

The strict partial preference order $\succ_\beta$ for the NHC with likelihood function lik on Θ_{G_N} corresponds to credal dominance for the NCC with as set Θ of parameters the likelihood-based confidence region $\{\theta \in \Theta_{G_N} : lik(\theta) > \beta\}$. When the NCC is learned from training data, the set Θ of parameters is usually estimated on the basis of the imprecise Dirichlet model: this model depends on a hyperparameter $s \in]0, +\infty[$, and the behavior of the resulting predictions as s varies from 0 to $+\infty$ is similar to the behavior as β varies from 1 to 0 of the predictions with cutoff point β returned by the NHC learned from the same training data. Besides the theoretical advantages of not needing prior distributions and of having the whole information encoded in the model (whereas to each $s \in]0, +\infty[$ corresponds a different NCC), the main practical advantage of the NHC over the NCC when they are learned from training data is that, unlike the hyperparameter s, the cutoff point β has a

frequentist interpretation in terms of (approximate) confidence levels, thanks to the result of (Wilks, 1938), as shown in the next example. A much more thorough comparison of these naive classifiers will be presented in (Antonucci et al., 2011).

Example 2. The simplest nontrivial classification problem corresponds to the case with $\Omega_C = \{a, b\}$ and $m = 0$. Assume that $P(C = a) = \frac{1}{2}$; in this case, the vacuous prediction of C can be considered as the theoretically correct classification, since there is no reason for preferring either of the two possible classes to the other. Consider the NHC learned from a complete training dataset consisting of n objects, and consider the NCC learned from the same training data on the basis of the imprecise Dirichlet model with the standard choice $s = 2$ for the hyperparameter. The probability that the prediction of C with cutoff point $\beta = 0.15$ returned by the NHC is vacuous is approximately 94.3% when $n = 100$ and 94.6% when $n = 1000$, while the probability that the prediction of C returned by the NCC is vacuous is approximately 23.6% when $n = 100$ and 7.6% when $n = 1000$. Hence, in this perfectly symmetric situation the probability that the NCC returns the vacuous prediction (that is, the theoretically correct classification) decreases as the number of objects in the training dataset increases.

5 Conclusion

When the likelihood function for the probabilities of a Bayesian network factorizes in multinomial likelihood functions, Theorem 1 gives a method for calculating profile likelihood functions for a particular class of probabilistic inferences. In the future, this method will be generalized to non-factorizing likelihood functions and more general classes of probabilistic inferences, by combining it with approximation algorithms (such as the EM algorithm) and exploiting the algebraic structure of the likelihood functions. Another interesting research topic is the combination of these methods with the learning of the graph of the Bayesian network.

Acknowledgments

The author wishes to thank Alessandro Antonucci for stimulating discussions on imprecise probabilistic graphical models, and the anonymous referees for their helpful comments.

References

Alessandro Antonucci and Marco Zaffalon. 2008. Decision-theoretic specification of credal networks: A unified language for uncertain modeling with sets of Bayesian networks. *Int. J. Approx. Reasoning*, 49(2):345–361.

Alessandro Antonucci, Marco E. G. V. Cattaneo, and Giorgio Corani. 2011. The naive hierarchical classifier. *In preparation.*

Marco E. G. V. Cattaneo. 2007. *Statistical Decisions Based Directly on the Likelihood Function.* Ph.D. thesis, ETH Zurich.

Marco E. G. V. Cattaneo. 2008. Fuzzy probabilities based on the likelihood function. In *Soft Methods for Handling Variability and Imprecision*, pages 43–50. Springer.

Marco E. G. V. Cattaneo. 2009. A generalization of credal networks. In *ISIPTA '09*, pages 79–88. SIPTA.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc., Ser. B*, 39(1):1–38.

Richard O. Duda and Peter E. Hart. 1973. *Pattern Classification and Scene Analysis.* Wiley.

Finn V. Jensen and Thomas D. Nielsen. 2007. *Bayesian Networks and Decision Graphs.* Springer, second edition.

Peter Walley. 1991. *Statistical Reasoning with Imprecise Probabilities.* Chapman & Hall.

Peter Walley. 1996. Inferences from multinomial data: Learning about a bag of marbles. *J. R. Stat. Soc., Ser. B*, 58(1):3–57.

Samuel S. Wilks. 1938. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *Ann. Math. Stat.*, 9(1):60–62.

Marco Zaffalon. 2002. The naive credal classifier. *J. Stat. Plann. Inference*, 105(1):5–21.

On a Discrete Dirichlet Model

Arthur Choi and Adnan Darwiche
University of California, Los Angeles
{*aychoi, darwiche*}*@cs.ucla.edu*

Abstract

The Dirichlet distribution is a statistical model that is deeply entrenched in the theory and practice of learning and reasoning in Bayesian networks. While it is mathematically convenient in certain situations, it imposes computational challenges in others, such as in Bayesian parameter learning under incomplete data. We consider in this paper a discretized variant of the continuous Dirichlet distribution. We show first how one can model such a distribution compactly as a Bayesian network, which can be used as a submodel in other Bayesian networks. We analyze some of its theoretical properties, relating the discrete variant to the original continous density. We further show how to represent and perform exact inference efficiently in this model. We finally discuss some implications that a discrete Dirichlet model may have in enabling the design of more sophisticated models, and in enabling new ways to reason about them.

1 Introduction

Bayesian networks are prevalent in the artificial intelligence and computer science communities, and in these domains, the natural model is often the one over discrete variables. Driven by the need to reason in these models, a significant body of research has developed, giving rise to effective and powerful algorithms for both exact and approximate inference in discrete Bayesian networks (Darwiche, 2009).

In contrast, for the task of learning Bayesian networks, one typically requires continuous variables representing, for example, possible network parametrizations. The Dirichlet is a popular model for such distributions, and has certainly been the most influential distribution in Bayesian learning (DeGroot, 1970; Heckerman, 1998). While they can be mathematically convenient in certain cases, the use of the Dirichlet distributions can pose a computational barrier in other situations. In particular, the Dirichlet distribution is the conjugate prior for the parameters of a multinomial distribution, and in the task of Bayesian parameter learning, a Dirichlet prior leads to a Dirichlet posterior, given complete data. However, in the case of incomplete data, the posteriors are in general no longer Dirichlet. To compute these posteriors, we must marginalize over the hidden variables, leading to a mixture of Dirichlets, which is both analytically and computationally prohibitive.[1] In such cases, we may appeal to variational approximations or Gibbs sampling, or otherwise settle for maximum a Posteriori (MAP) parameter estimates (Heckerman, 1998).

Considering the vast body of research on reasoning in discrete models, and considering further the increasing availability of computational resources (in the form of many-core and massively parallel architectures, distributed computing, cloud computing, etc.), posing such problems in fully-discrete approximations may become a compelling alternative. Towards this end, we consider in this paper a discretized variant of the Dirichlet distribution. A naive discretization of this domain would enumerate a set of candidate distributions, which may be coming from a high-dimensional space. In con-

[1] Drawing samples from a Dirichlet distribution is another difficulty, where in practice, approaches based on rejection sampling are often used. This is another case where a discrete Dirichlet has been considered as an alternative to the continuous one (Matsui et al., 2010).

Pp. 65–73 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

trast, we propose a natural but compact representation of this domain that can be encoded directly as a Bayesian network, allowing it to be embedded naturally in other Bayesian network models. We further show that this discrete Dirichlet sub-model is further amenable to exact inference, via a specialized belief propagation procedure which we describe.

We also analyze the theoretical properties of this discrete Dirichlet model, relating it to the original continuous distribution. We conclude by discussing some of the advantages, in terms of both modeling and reasoning, that present themselves by assuming a discrete representation of Dirichlet priors.

2 Preliminaries

Let X be a random variable taking on k possible values x_i. Let the distribution over X be parametrized by a vector $\theta_X = (\theta_{x_1}, \ldots, \theta_{x_k})$, where each θ_{x_i} is a probability and $\sum_i \theta_{x_i} = 1$. We shall refer to θ_X as a parameter set and each θ_{x_i} as a parameter. Given a specific parameter set θ_X we thus have the probability $Pr(X = x_i \mid \theta_X = (\theta_{x_1}, \ldots, \theta_{x_k})) = \theta_{x_i}$. Note that when the symbol θ_{x_i} appears in the context of θ_X, it refers to the i-th component of the parameter set θ_X .

The Dirichlet distribution is the one typically used to specify a prior probability density over parameter sets θ_X:

$$\rho(\theta_X = (\theta_{x_1}, \ldots, \theta_{x_k})) = \eta \prod_{i=1}^{k} [\theta_{x_i}]^{\psi_{x_i}-1} \quad (1)$$

where the exponents ψ_{x_i} are hyper-parameters of the Dirichlet, and η is a normalizing constant:

$$\eta = 1/\int \prod_{i=1}^{k} [\theta_{x_i}]^{\psi_{x_i}-1} \, d\theta_X = \frac{\Gamma(\sum_{i=1}^{k} \psi_{x_i})}{\prod_{i=1}^{k} \Gamma(\psi_{x_i})}$$

where Γ is the gamma function.

We next present a discrete approximation of the Dirichlet distribution and discuss some of its properties and advantages in later sections.

3 A Discretized Dirichlet

Suppose we discretize the space of a parameter set $\theta_X = (\theta_{x_1}, \ldots, \theta_{x_k})$ so that each parameter

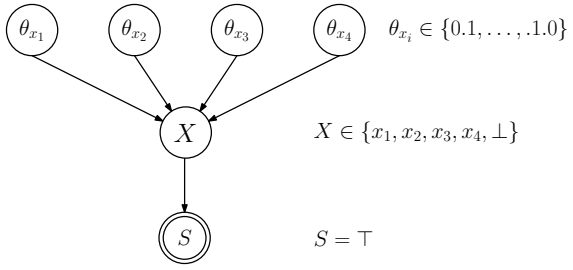


Figure 1: A Bayesian network model for a discrete Dirichlet distribution.

θ_{x_i} takes a value from a finite set Ω_{x_i} of probabilities. Let Ω_X be the values of parameter set $\theta_X = (\theta_{x_1}, \ldots, \theta_{x_k})$ such that each $\theta_{x_i} \in \Omega_{x_i}$ and $\theta_{x_1} + \ldots + \theta_{x_k} = 1$. We can now define a discrete analogue of the Dirichlet distribution:

$$Pr(\theta_X = (\theta_{x_1}, \ldots, \theta_{x_k})) = \beta \prod_{i=1}^{k} [\theta_{x_i}]^{\psi_{x_i}-1} \quad (2)$$

for $\theta_X \in \Omega_X$ where β is a normalizing constant. The exponents ψ_{x_i} are the hyper-parameters of the now discrete Dirichlet distribution.[2]

3.1 A Bayesian Network Micro-Model

We present now a Bayesian network micro-model of the discrete analogue of the Dirichlet we just described. This network fragment can be embedded in any other discrete Bayesian network model, and may be augmented further as dictated by any prior knowledge available.

Let X be a variable with k values $x_1, \ldots, x_k$, and that we want to specify a prior over its distribution. Figure 1 illustrates an example of our model. We model each parameter θ_{x_i} of the parameter set $\theta_X = (\theta_{x_1}, \ldots, \theta_{x_k})$ as marginally independent root variables. These parameter variables serve as the prior for the CPT parameters of variable X. We include another observed variable S that enforces a constraint that the parameter set θ_X normalizes to sum to one.

[2] Note that others have also investigated discretizations of the Dirichlet for other purposes, such as (Matsui et al., 2010), who were interested in drawing samples from a discrete Dirichlet. In contrast, we propose next an explicit Bayesian network representation, and in Section 5 we provide an exact inference algorithm for it.

First, each root variable θ_{x_i} has values in Ω_{x_i}, having the following CPT:

$$Pr(\theta_{x_i}) = \alpha_{x_i}[\theta_{x_i}]^{\psi_{x_i}-1},$$

where α_{x_i} is a normalizing constant:

$$\alpha_{x_i} = 1/\sum_{\theta_{x_i}\in\Omega_{x_i}} [\theta_{x_i}]^{\psi_{x_i}-1}$$

Next, variable X has values in $\{x_1, \ldots, x_k, \bot\}$, where $\bot$ is a new distinguished state capturing invalid distributions that do not sum to one. Its CPT is as follows:

- If $\theta_{x_1} + \ldots + \theta_{x_k} = 1$, then

$$Pr(X{=}x_i \mid \theta_{x_1}, \ldots, \theta_{x_k}) = \theta_{x_i} \text{ for all } x_i$$
$$Pr(X{=}\bot \mid \theta_{x_1}, \ldots, \theta_{x_k}) = 0$$

- If otherwise $\theta_{x_1} + \ldots + \theta_{x_k} \neq 1$, then

$$Pr(X{=}x_i \mid \theta_{x_1}, \ldots, \theta_{x_k}) = 0 \text{ for all } x_i$$
$$Pr(X{=}\bot \mid \theta_{x_1}, \ldots, \theta_{x_k}) = 1$$

Finally, variable S has two values: $\top$ representing valid parameter sets, and $\bot$ representing invalid parameter sets. Its CPT is as follows:

$$\begin{aligned} Pr(S{=}\top \mid X{=}x_i) &= 1 \\ Pr(S{=}\bot \mid X{=}x_i) &= 0 \end{aligned}$$

for all x_i, and

$$\begin{aligned} Pr(S{=}\top \mid X{=}\bot) &= 0 \\ Pr(S{=}\bot \mid X{=}\bot) &= 1 \end{aligned}$$

The observation $S{=}\top$ represents a constraint that parameter sets θ_X must be valid, forcing invalid parameter sets to have probability zero.

In the extended paper, we prove the following key property of our proposed micro-model:

$$Pr(\theta_{x_1}, \ldots, \theta_{x_k} \mid S{=}\top) = \beta \prod_{i=1}^{k} [\theta_{x_i}]^{\psi_{x_i}-1} \quad (3)$$

which is precisely the distribution given in Equation 2. This basically shows that once we condition our micro-model on the observation $S{=}\top$, the model induces a discrete Dirichlet distribution on parameter sets. We examine the properties of this distribution further in Section 4, where we show how it resembles the continuous Dirichlet distribution in key aspects.

Before we proceed, we make some observations. First, observe that a naive discretization would enumerate a finite set of parameter sets $\theta_X = \{\theta_{x_1}, \ldots, \theta_{x_k}\}$, which require an intractably large selection to achieve a reasonable coverage of the domain Ω_X. Although our variable X has k parents, one for each parameter θ_{x_i}, and the CPT has a number of entries that is exponential in k, our representation remains compact as we may leave the CPT of X defined implicitly. Moreover, as our micro-model has a polytree structure, we can in fact perform inference efficiently in this implicit representation, as we show in Section 5.

3.2 An Example

Consider again our example from Figure 1. We have assumed a random variable X with four states $x_1, \ldots, x_4$, and a distribution over X has four parameters in its parameter set: $\theta_X = (\theta_{x_1}, \ldots, \theta_{x_4})$. Suppose then that each parameter can take on one of the $n = 10$ values in $\theta_{x_i} = \{0.1, 0.2, \ldots, 1.0\}$. Indeed, we can think of $\frac{1}{n}$ as a discretization granularity, and define parameter domains such as $\Omega_{x_i} = \{\frac{a}{n} \mid a \in \{1, 2, \ldots, n\}\}$. Using an exponent $\psi_{x_i} = 1$, we have the prior distribution over parameter values $Pr(\theta_{x_i}{=}p) = 0.1$ for all $p \in \Omega_{x_i}$. Using an exponent $\psi_{x_i} = 2$, we have the prior distribution $Pr(\theta_{x_i}{=}p) = \alpha \cdot p$ where $\alpha = \sum_{a=1}^{10} \frac{a}{10} = 5.5$.

4 Properties of the Discrete Dirichlet

In this section, we examine the properties of the discrete Dirichlet distribution of Equation 3, comparing it to the original continuous distribution of Equation 1. In particular, one of our goals here is to confirm that the discrete Dirichlet distribution behaves in a natural way, as compared to the continuous version that it is based on. When we make use of the original continuous Dirichlet, we refer to a density ρ,

and when we make use of the discrete version, we refer to a distribution Pr.

Expected Value. The first property of our micro-model is its ability to explicitly represent the expected value of each parameter θ_{x_j}, which is known as the *Bayesian estimate* (Heckerman, 1998). In particular, we prove the following in the extended paper:

$$\begin{aligned}\text{Ex}[\theta_{x_j}] &= Pr(X{=}x_j \mid S{=}\top) \\ &= \sum_{\theta_X \in \Omega_X} \theta_{x_j} \cdot Pr(\theta_X \mid S{=}\top)\end{aligned}$$

which shows that we can recover the Bayesian estimates of our parameter set θ_X directly from the distribution of variable X conditioned on observation $S{=}\top$.

Expected Value: An Exact Case. In general, the expected value of a parameter θ_{x_j} from the discrete Dirichlet is only an approximation for that of the continuous Dirichlet. However, in the special case that the Dirichlet exponents are all the same exponent ψ, the expected value is the same under both distributions:

$$\text{Ex}[\theta_{x_j}] = \frac{\psi_{x_j}}{\sum_{i=1}^{k} \psi_{x_i}} = \frac{\psi}{k \cdot \psi} = \frac{1}{k}$$

which yields a parameter set θ_X that is uniform.[3] Note that this holds even in the case the discretization does not admit a uniform parameter set (i.e., $\frac{1}{k} \notin \Omega_{x_j}$); we will see an example of this in Section 5.1.

Observed Data. Suppose now that we have a data set $\mathcal{D}$ where we have observed N cases, with N_i cases observed for each value x_i of variable X. In the continuous case, we have the posterior Dirichlet:

$$\rho(\theta_X \mid \mathcal{D}) \propto \prod_{i=1}^{k} [\theta_{x_i}]^{N_i + \psi_{x_i} - 1}$$

which is a Dirichlet with exponents $N_i + \psi_{x_i}$. In the discretized case, assume that we have replicated instances of the X variable, each sharing the same parents in parent set θ_X. In the extended paper, we show the following for the discrete case

$$Pr(\theta_X \mid \mathcal{D}) \propto \prod_{i=1}^{k} [\theta_{x_i}]^{N_i + \psi_{x_i} - 1}$$

which is a discrete Dirichlet with exponents $N_i + \psi_{x_i}$, therefore, resembling the continuous distribution in this case.

Remember at this point that if we were using the Dirichlet for Bayesian parameter learning under incomplete data, the posterior is in general not Dirichlet. Analogously, if we had used a discrete Dirichlet. On the other hand, the posterior in this latter case is still a discrete distribution, which leaves us with more options in terms of performing exact and approximate inference, as we shall discuss in the next section.

5 Inference in a Discrete Dirichlet

In Section 3, we proposed an efficient representation for a discrete Dirichlet distribution, assuming that the CPT of variable X is implicitly defined. Taking advantage of the fact that the network fragment is a polytree, and that we can leave the CPT of X implicit, we propose a belief propagation (BP) algorithm for exact inference in our sub-model. The corresponding message update equations can then be used in a general belief propagation procedure for performing (possibly approximate) inference in a Bayesian network with discrete Dirichlet submodels. The inference procedure we describe may also be used in other inference frameworks, which we will discuss later in this section.

Our presentation will be similar in spirit to inference using noisy-or CPTs. The noisy-or model also has a compact representation that is linear in the number of parents, and has an efficient belief propagation procedure for performing inference (Pearl, 1988, Section 4.3.2).

First, consider the following forms for the message updates required for performing belief propagation (BP) (Pearl, 1988):

$$\pi_X(\theta_{x_j}) = Pr(\theta_{x_j}) \tag{4}$$

$$\lambda_X(\theta_{x_j}) = \sum_{(\theta_{x_1},\ldots,\theta_{x_j},\ldots,\theta_{x_k}) \in \Omega_X} \prod_{i \neq j} \pi_X(\theta_{x_i}) \tag{5}$$

[3] To sketch the proof, we note that if $\psi_{x_i} = \psi_{x_j}$, then θ_{x_i} and θ_{x_j} are not otherwise distinguishable, so it must be that the distribution $Pr(\theta_{x_i} \mid S{=}\top)$ is equivalent to the distribution $Pr(\theta_{x_j} \mid S{=}\top)$. By Equation 7, it follows that $\text{Ex}[\theta_{x_i}] = \text{Ex}[\theta_{x_j}]$ for all x_i, x_j.

Here, $\pi_X(\theta_{x_j})$ is the message passed from parameter node θ_{x_j} to its child X, and $\lambda_X(\theta_{x_j})$ is the message that X passes to its parent θ_{x_j}. Using these messages, we can compute the posterior marginals:

$$Pr(\theta_{x_j}|S=\top) \propto \pi_X(\theta_{x_j})\lambda_X(\theta_{x_j}) \quad (6)$$

$$Pr(X=x_j|S=\top) = \sum_{\theta_{x_j}\in\Omega_{x_j}} \theta_{x_j} Pr(\theta_{x_j}|S=\top) \quad (7)$$

The key computational component in this procedure is for the message $\lambda_X(\theta_{x_j})$, which requires an efficient evaluation of terms of the following form:

$$f(I,p) = \sum_{\theta_{X_I}\in\Omega^p_{X_I}} \prod_{i\in I} \pi_X(\theta_{x_i}) \quad (8)$$

Here, I is an index set $I \subseteq \{1,\ldots,k\}$ that selects a subset of the states x_i of variable X. Moreover, parameter set θ_{X_I} contains the selection of parameters θ_{x_i} for each index $i \in I$. Finally, $\Omega^p_{X_I}$ denotes the domain of parameter sets θ_{X_I} that sum to p, i.e., $\theta_{X_I} \in \Omega^p_{X_I}$ iff $\sum_{i\in I}\theta_{x_i} = p$. For the case of Equation 5, $I = \{1,\ldots,k\} \setminus j$ and $p = 1 - \theta_{x_j}$.

We sketch in Appendix A how to compute these summations efficiently. More specifically, suppose that we have k parameters θ_{x_k}, and we have n possible parameter values, i.e., $|\Omega_{x_i}| = n$. We sketch in the Appendix how all messages $\lambda_X(\theta_{x_j})$ can be computed in time $O(k \cdot n^2)$, which is polynomial and avoids the exponential (in k) computation required by standard belief propagation. In an extended version of the paper, we show how one can compute BP messages when this model is embedded in a network where X has parents and children. The computation of these messages are similar in spirit, and also rely primarily on Equation 8.

To conclude this section, we remark that the inference equations we have identified in this section (together with more general ones in an extended version) can be used to perform inference in a general Bayesian network that has discrete Dirichlet sub-models embedded in it. If this Bayesian network is also a polytree, the equations can be used to perform exact inference using belief propagation, where we apply updates according to Equations 4 and 5 for messages passed along edges in our discrete Dirichlet sub-model. Analogously, we can perform approximate inference in a network that is not a polytree, by using loopy belief propagation (Yedidia et al., 2003). These exact computations may also be used in approximate inference frameworks based on performing exact inference in approximate networks, such as variational approximations (Jaakkola, 2001), and generalizations of belief propagation based on structural relaxations (Choi and Darwiche, 2006; Choi and Darwiche, 2009). The latter approach, in particular, could assume a structural relaxation where discrete Dirichlet sub-models are independent (where the relaxation is later compensated for). In such an approach, one needs only to perform exact inference independently in each discrete Dirichlet sub-model.

5.1 Examples

Consider again our example from Figure 1 and Section 3.2, where we are now interested in the distribution $Pr(\theta_X \mid S=\top)$ over parameter sets, and the expected parameter values $\mathrm{Ex}[\theta_X]$ implied by it. Assuming we have Dirichlet exponents $\psi_{x_i} = 1$ for all four x_i, we have the following distribution over parameter values:

θ_{x_i}	$Pr(\theta_{x_i} \mid S=\top)$	θ_{x_i}	$Pr(\theta_{x_i} \mid S=\top)$
0.1	33.33%	0.6	3.57%
0.2	25.00%	0.7	1.19%
0.3	17.86%	0.8	0.00%
0.4	11.90%	0.9	0.00%
0.5	7.14%	1.0	0.00%

for all four x_i. Note that since each parameter θ_{x_i} must be at least 0.1, and there are four parameters θ_{x_i}, it is not possible for a parameter to have a value of 0.8, 0.9 or 1.0. The expected parameter values (the Bayesian estimates) are:

$$\mathrm{Ex}[\theta_{x_i}] = \sum_{\theta_X\in\Omega_X} \theta_{x_i} \cdot Pr(\theta_X \mid S=\top) = 0.25$$

for all x_i, which is the uniform distribution, and also the expected parameter values given by the original continuous Dirichlet.

As another example, if we have Dirichlet exponents $\psi_{x_i} = 2$, we have the following distribution over parameter values:

θ_{x_i}	$Pr(\theta_{x_i} \mid S=\top)$	θ_{x_i}	$Pr(\theta_{x_i} \mid S=\top)$
0.1	26.92%	0.6	2.10%
0.2	29.37%	0.7	0.41%
0.3	22.03%	0.8	0.00%
0.4	13.05%	0.9	0.00%
0.5	6.12%	1.0	0.00%

and again we have the parameter estimates $\text{Ex}[\theta_X] = (25\%, 25\%, 25\%, 25\%)$, as would be given by the continuous Dirichlet. Since this is a small example, we can also compute the MAP estimates $\text{argmax}_{\theta_X} Pr(\theta_X \mid S=\top)$, using a generic MAP algorithm, such as (Park and Darwiche, 2004). For the continuous Dirichlet, the expected value $\text{Ex}[\theta_X]$ with respect to density $\rho(\theta_X)$ is equivalent to the MAP estimates $\text{argmax}_{\theta_X} \rho(\theta_X)$, in this case. The discrete Dirichlet's MAP estimates are not unique here: there are $\binom{4}{2} = 6$ MAP estimates for the discrete Dirichlet, each having two parameters $\theta_{x_i} = 20.0\%$ and two parameters $\theta_{x_j} = 30.0\%$. This is due, however, to the particular discretization we used which cannot represent a uniform distribution. If we use, e.g., 20 discrete states, the discrete Dirichlet has a unique and uniform MAP estimate.

Suppose now we have exponents $(1, 2, 3, 4)$. The continuous Dirichlet yields expected parameter values $\text{Ex}[\theta_X] = (10\%, 20\%, 30\%, 40\%)$. Varying the number of discrete states n used in our discretization, we arrive at the following, increasingly accurate parameter estimates from the discrete Dirichlet (compared to the continuous ones):

n=10	(15.05%, 20.11%, 27.86%, 36.98%)
n=20	(12.38%, 19.77%, 29.08%, 38.77%)
n=50	(10.92%, 19.84%, 29.67%, 39.56%)
n=100	(10.46%, 19.91%, 29.84%, 39.79%)
n=1000	(10.05%, 19.99%, 29.98%, 39.98%)

Note that by $n = 47$ (not shown), the maximum absolute error is less than 1%.

Consider now a network $\theta_X \rightarrow X \rightarrow Z$ where θ_X is a continuous Dirichlet, and where we have observed $Z=z$. Note that if X is unobserved, the posterior $\rho(\theta_X \mid Z=z)$ is in general only a mixture of Dirichlet's, which are generally unwieldy, both analytically and computationally (Heckerman, 1998). In contrast, if we represent this network using our discrete Dirichlet, each variable θ_{x_i} in our parameter set is just another discrete variable in a discrete Bayesian network, which we can approach using many of the tools and algorithms available to us, such as belief propagation, as we described earlier. In this sense, the discrete Dirichlet yields an approximation to the above mixture of Dirichlet's.

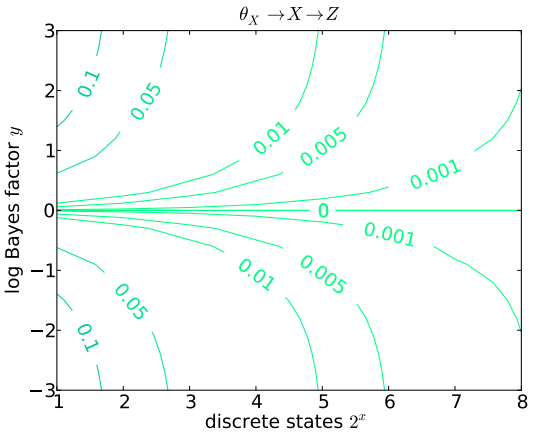


Figure 2: The error introduced by discretization. On the x-axis, we vary the number of discrete states (in powers of 2), and on the y-axis we vary the strength of the link $X \rightarrow Z$.

We evaluate the quality of this approximation in Figure 2, which depicts the absolute maximum error seen in the discrete Dirichlet's parameter estimates as compared to those obtained from the mixture of continuous Dirichlets. We assume here that variable X has two states and that we have observed only $Z=z$. We vary: (1) on the horizontal axis, the number of discrete states n used for the parameter domains $\Omega_{x_i} = \{\frac{a}{n} \mid a \in \{1, 2, \ldots, n\}\}$; and (2) on the vertical axis, the strength of link $X \rightarrow Z$:

$$\log \frac{Pr(Z=z|X=x_1)}{Pr(Z=z|X=x_2)}$$

which is the log Bayes factor for the event $X=x_1$ and observation $Z=z$. Note that for the continuous Dirichlet, we have only one hidden variable X, so it is still tractable to enumerate over all cases $X=x_i$ to compute $\rho(\theta_X \mid Z=z)$.

In Figure 2, we plot the contours where the number of discrete states n and the log Bayes

factor yield a maximum absolute error of E, for different errors E. We used Dirichlet exponents $\psi_{x_1} = 1$ and $\psi_{x_2} = 1$. We observe a few general trends. First, as we increase the number of discrete states n, we find the error decreases, as expected. Second, as we increase the strength of the link $X \rightarrow Z$, we find that the error tends to increase. Third, if the link $X \rightarrow Z$ is vacuous, the discrete Dirichlet's parameter estimates are exact (they recover the uniform distribution). We note also that using only $2^5 = 32$ discrete states, the errors E for the range of y considered are below 1%.

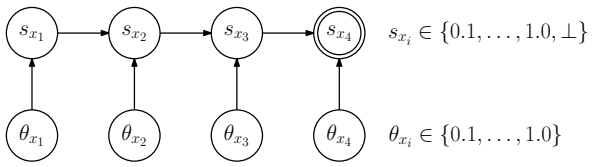


Figure 3: Another micro-model, enforcing the constraint that parameters θ_{x_i} sum to one. We maintain the cumulative sum of the parameters θ_{x_i} in the variables s_{x_i}. We clamp the end of the chain s_{x_k} to 1 as evidence. The state $\perp$ now indicates a sum that has surpassed the value 1.

6 Discussion

Continuous distributions (such as Dirichlet and logistic normal distributions) have been an integral part of learning Bayesian networks. The use of continuous distributions, however, can limit both the scalability and representational power of these models. On scalability, these distributions constrain the class of algorithms one can use to perform learning and inference with the models. On the representational side, they provide restrictions on what can be modeled as the result must fit into one of the known distributions (such as Dirichlet).

A sound and principled procedure for designing purely, or more, discrete models could potentially broaden the use and scope of learning. Topics models are a particularly relevant example (Blei and Lafferty, 2009), where there is significant interest in augmenting and designing new models, to enable new analysis and queries. One of the challenges, however, is that one generally needs to design new algorithms for learning and inference when one is dealing with a new or augmented model. In contrast, consider two points: (1) practitioners are already in general well-versed in discrete modeling, and would more easily be able to incorporate prior knowledge for their particular applications (Niculescu, 2005), and (2) there is a great body of research devoted to reasoning in discrete Bayesian networks that we can immediately take advantage of.

In this paper, we have laid some of the groundwork for a discrete model of the Dirichlet distribution, targeting the longer-term goals of developing compelling alternatives for learning and modeling Bayesian networks. Given the Bayesian network micro-model for the Bayesian network, and the exact and efficient algorithm for reasoning in it, we are now in a position to start developing new learning algorithms, formulated in terms of performing (approximate) inference in a meta-network for Bayesian parameter learning (Darwiche, 2009), as we discussed in Section 5.

A Inference: Proof Sketch

We sketch here how to efficiently compute the messages $\lambda_X(\theta_{x_j})$ of Equation 5, which is the central computational component for performing exact inference in the discrete Dirichlet submodel. Our approach is based on message passing in an augmented network where every node has at most two parents. Consider first a network where we marginalize out variable X:

$$\begin{aligned} Pr(S{=}\top \mid \theta_X) &= \sum_X Pr(S{=}\top \mid X) Pr(X \mid \theta_X) \\ &= \sum_{X \neq \perp} Pr(X \mid \theta_X) \end{aligned}$$

since $Pr(S{=}\top \mid X{=}\perp) = 0$, and 1 otherwise for all $X{=}x_i$. If $\theta_X \in \Omega_X$ (it sums to one), then

$$\sum_{i=1}^{k} Pr(X{=}x_i \mid \theta_X) = \sum_{i=1}^{k} \theta_{x_i} = 1$$

and zero otherwise (when $\theta_X \notin \Omega_X$). Variable S now has parameter nodes θ_{x_i} as direct parents. We now augment the variable S, which enforces the sum-to-one constraint, into a chain that enforces this constraint by accumulating the sum of the parameters θ_{x_i}; see Figure 3. Here, $Pr(s_{x_i} \mid s_{x_{i-1}}, \theta_{x_i}) = 1$ iff $s_{x_i} = s_{x_{i-1}} + \theta_{x_i}$, and $Pr(s_{x_1} \mid \theta_{x_1}) = 1$ iff $s_{x_1} = \theta_{x_1}$.

Consider now a message passed from s_{x_i} to $s_{x_{i+1}}$, for some $1 < i \leq k$:

$$\begin{aligned}
&\pi_{s_{x_{i+1}}}(s_{x_i}) \\
&= \sum_{s_{x_{i-1}}} \sum_{\theta_{x_i}} Pr(s_{x_i} | s_{x_{i-1}}, \theta_{x_i}) \pi_{s_{x_i}}(s_{x_{i-1}}) \pi_{s_{x_i}}(\theta_{x_i}) \\
&= \sum_{s_{x_{i-1}} + \theta_{x_i} = s_{x_i}} \pi_{s_{x_i}}(s_{x_{i-1}}) \pi_{s_{x_i}}(\theta_{x_i}) \qquad (9)
\end{aligned}$$

since $Pr(s_{x_i} \mid s_{x_{i-1}}, \theta_{x_i}) = 0$ if $s_{x_{i-1}} + \theta_{x_i} \neq s_{x_i}$. By recursively substituting this equation for messages $\pi_{s_{x_i}}(s_{x_{i-1}})$, we find that:

$$\pi_{s_{x_{i+1}}}(s_{x_i}) = \sum_{\theta_{x_1} + \cdots + \theta_{x_i} = s_{x_i}} \prod_{j=1}^{i} \pi_{s_{x_j}}(\theta_{x_j})$$

which is a summation of the form in Equation 8. We can then compute Equation 8 for a given I and p by permuting the variables in our network so that indices I appear at the head of the chain. To compute all of the messages of Equation 5, however, we need only perform message-passing once in the network of Figure 3, since one can show that the messages $\lambda_{s_{x_i}}(\theta_{x_i})$ will be the messages $\lambda_X(\theta_{x_i})$ that we desire.

Computing a message takes (at most) $O(n)$ time for each of its $O(n)$ entries (as in Equation 9). There are $2k - 1$ edges in this model, so to compute all messages of Equation 8, we require only $O(k \cdot n^2)$ time.

References

David M. Blei and John D. Lafferty. 2009. Topic models. In Ashok Srivastava and Mehran Sahami, editors, *Text Mining: Classification, Clustering, and Applications*, chapter 4, pages 71–93. Chapman and Hall/CRC.

Arthur Choi and Adnan Darwiche. 2006. An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 1107–1114.

Arthur Choi and Adnan Darwiche. 2009. Relax then compensate: On max-product belief propagation and more. In *Proceedings of the Twenty-Third Annual Conference on Neural Information Processing Systems (NIPS)*, pages 351–359.

Adnan Darwiche. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Morris H. DeGroot. 1970. *Optimal Statistical Decisions*. McGraw-Hill.

David Heckerman. 1998. A tutorial on learning with Bayesian networks. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 301–354. MIT Press.

Tommi Jaakkola. 2001. Tutorial on variational approximation methods. In D. Saad and M. Opper, editors, *Advanced Mean Field Methods*, chapter 10, pages 129–160. MIT Press.

Tomomi Matsui, Mitsuo Motoki, Naoyuki Kamatani, and Shuji Kijima. 2010. Polynomial time approximate or perfect samplers for discretized Dirichlet distribution. *Japan Journal of Industrial and Applied Mathematics*. To appear (currently published online).

Radu Stefan Niculescu. 2005. *Exploiting Parameter Domain Knowledge for Learning in Bayesian Networks*. Ph.D. thesis, Carnegie Mellon University.

James Park and Adnan Darwiche. 2004. A differential semantics for jointree algorithms. *Artificial Intelligence*, 156:197–216.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. 2003. Understanding belief propagation and its generalizations. In Gerhard Lakemeyer and Bernhard Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*, chapter 8, pages 239–269. Morgan Kaufmann.

Optimizing the triangulation of Dynamic Bayesian Networks

Morgan Chopin and Pierre-Henri Wuillemin
LIP6
University of Paris 6
75016, Paris, France

Abstract

In this paper, we address the problem of finding good quality elimination orders for triangulating dynamic Bayesian networks. In Bilmes and Bartels (2003), the authors proposed a model and an algorithm to compute such orders, but in exponential time. We show that this can be done in polynomial time by casting the problem to the problem of finding a minimum s–t cut in a graph. In this approach, we propose a formal definition of an *interface* (a set of nodes which makes the past independent from the future), we link the notion of an interface with the notion of a graph cut-set. We also propose an algorithm which computes the minimum interface of a dBN in polynomial time. Given this interface, we show how to get an elimination order which guarantees, theoretically and experimentally, the triangulation quality.

1 Introduction

Many domains deal with monitoring the evolution of complex system over time: localization of a robot in a dynamic environment (Fox et al., 1999), fault diagnostics analysis (Weber, 2002), monitoring physicochemical reactions (Baudrit et al., 2009). Many formal tools have been developed for describing such systems, including *hidden Markov models*, *Kalman filters*, and *dynamic Bayesian networks* (dBNs) (Dean and Kanazawa, 1990). All of these models share the same idea of representing the state of a system and its changes over time according to a probabilistic *transition model* and a *prior* distribution. Moreover, they assume the system to be *first-order Markovian* (the state at time t only depends on the state at time $t-1$) and *homogeneous* (the parameters do not vary with time). DBNs are graphical models that generalize hidden Markov models and Kalman filters by factoring the state as a *Bayesian network* (Pearl, 1988), hence providing an easy way to model and specify the parameters of a complex system. A dBN is then used to answer some *query* we may ask by making an *inference* (*i.e.* compute the probability of a state given some observations). Numerous types of queries lead to numerous types of inference. In this paper, we study the *offline* inference problem: computing the probability of a state at time $t \in [0, T]$ given the evidences we have during the simulation.

Exact inference in Bayesian networks is computationally hard (Cooper, 1990). This is even more difficult in the case of a dBN because it monitors variables over time and thus can be arbitrarily large, depending of the duration of the process. Among all the methods that perform inference in dBNs (Murphy, 2002), some are based on triangulation such as the junction tree inference (Jensen et al., 1990). Finding a good triangulation, *i.e.* finding an elimination order such that the resulting triangulated graph has the smallest maximum clique, is crucial for tractable inference but is a NP-complete problem (Arnborg et al., 1987). Hence, we must resort on heuristic approach to find good elimination orders.

In the case of dBNs the most promising work of finding good quality elimination orders uses a constrained elimination scheme (Kjærulff, 1994; Murphy, 2002; Darwiche, 2001; Bilmes and Bartels, 2003). In this approach, the use of constraints reduce the amount of elimination orders

Pp. 73–81 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

to consider. Furthermore, these constraints do not reduce the triangulation quality (Darwiche, 2001). In this paper, we propose a method that follows the same idea by computing such elimination orders in an efficient[1] way. We also give some theoretical and experimental results.

Section 2 gives the basic background and defines the notion of interface. In section 3, we present the state of the art inference methods in a dBN and their complexity. In section 4, we present our method. Experimental results are given in section 5 and we conclude in section 6. Throughout this paper, we assume that the reader has some knowledge in graphical models (Pearl, 1988).

2 Dynamic Bayesian networks

We begin by defining some notations. Discrete random variables will be denoted by X^1, X^2,... and X_t^i will denote the i-th variable at time t. We will use the notation $X_{a:b}$ as a shortcut for $X_a, ..., X_b$. In a Bayesian network, we will denote by $pa(X)$ the parents of a node X in the graph, *i.e.* the set of nodes with an arc to X. A set of random variables will be denoted by $\mathbf{X}$; its size, denoted $size(\mathbf{X})$, corresponds to the cardinal product of each variable in $\mathbf{X}$ and its dimension, denoted $|\mathbf{X}|$, is the number of variable in $\mathbf{X}$. Finally, we will use $\mathbf{X}_t$ to denote the set of variables at time t.

Dynamic Bayesian networks (dBNs) can be viewed as an extension of Bayesian networks for modeling dynamic systems. Since we assume that the process is first-order Markovian and homogeneous, we can write the transition model as follows:

$$P(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = P(\mathbf{X}_t|\mathbf{X}_{t-1}) = P(\mathbf{X}_1|\mathbf{X}_0)$$

In addition, a dBN exploits independences into $\mathbf{X}_1$ and $\mathbf{X}_0$ to factorize $P(\mathbf{X}_1|\mathbf{X}_0)$ and $P(\mathbf{X}_0)$. This leads us to the following definition:

Definition 1. A 2-*Time-slice Bayesian Network* (2-TBN) is a Bayesian network defined as follows: the nodes are partitioned into two sets $\mathbf{X}_0$ and $\mathbf{X}_1$ called *slices*. The 2-TBN represents the *transition model* such that:

$$P(\mathbf{X}_1|\mathbf{X}_0) = \prod_{i=0}^{n-1} P(X_1^i|pa(X_1^i))$$

where $pa(X_1^i) \subset \mathbf{X}_0 \cup \mathbf{X}_1$ and the *prior* distribution:

$$P(\mathbf{X}_0) = \prod_{i=0}^{n-1} P(X_0^i|pa(X_0^i))$$

where $pa(X_0^i) \subset \mathbf{X}_0$.

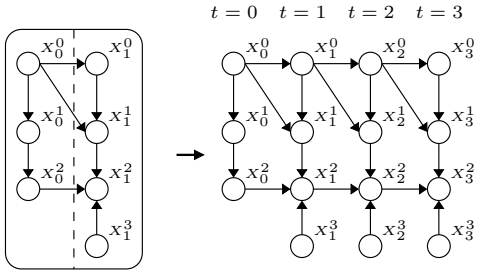


Figure 1: A 2-TBN unrolled three times to get a dBN of length $T = 3$. Here $\mathbf{I}_t^{\rightarrow} = \{X_t^0, X_t^2\}$ and $\mathbf{I}_t^{\leftarrow} = \{X_t^0, X_t^1, X_t^2\}$.

We can then define a dBN (see Figure 1):

Definition 2. A dynamic Bayesian network of length T (denoted as $\mathcal{G}_T = (\mathbf{X} = \mathbf{X}_0 \cup ... \cup \mathbf{X}_T, \mathbf{E})$) is a Bayesian network resulting by "unrolling" a 2-TBN in T time steps. Each state $\mathbf{X}_t$ of the dBN is called the slice t. We denote by $\mathcal{G}_T^m$ the *moralization* of the dBN $\mathcal{G}_T$.

In the next section, we define the notion of interface that plays a crucial role in our triangulation method.

2.1 The notion of interface

An interface is a subset of nodes such that if they were removed, the past would be independent from the future[2]. This notion admits several definitions (Kjærulff, 1994; Murphy, 2002; Darwiche, 2001; Bilmes and Bartels, 2003), we propose one that allows us to encompass several definitions found in the literature and makes explicit the link between an interface and a cut-set in a graph.

[1] By *efficient* we mean *polynomial time*.

[2] Note that we use the term interface to be consistent with the literature. Heggernes (2006) uses the term *separator*.

Definition 3. Let $\mathcal{G}_T^m = (\mathbf{X}, \mathbf{E})$ a moralized dBN of length T, a subset $\mathbf{I} \subset \mathbf{X}$ is called *interface* if every path from a node of $\mathbf{X}_0$ to a node of $\mathbf{X}_T$ in $\mathcal{G}_T^m$ contains at least one node in $\mathbf{I}$. A *minimal interface* $\mathbf{I}$ is such that for all $\mathbf{I}' \subset \mathbf{I}$, $\mathbf{I}'$ is not an interface.

Now we prove that forward and backward interfaces defined in Darwiche (2001) are also interfaces. First, we recall what are backward and forward interfaces:

Definition 4. Let $\mathcal{G}_T = (\mathbf{X}, \mathbf{E})$ a dBN of length T, the *forward interface* $\mathbf{I}_t^{\rightarrow}$ is the set of nodes in slice $t < T$ that have at least one child in slice $t+1$. The *backward interface* $\mathbf{I}_t^{\leftarrow}$ is the set of all nodes X in slice $t > 0$ such that X, or one of its children, has a parent in slice $t-1$.

Proposition 1. *Let $\mathcal{G}_T = (\mathbf{X}, \mathbf{E})$ a dBN of length T, then $\mathbf{I}_t^{\rightarrow}$ for all $t < T$ and $\mathbf{I}_{t'}^{\leftarrow}$ for all $t' > 0$ are interfaces.*

Proof. Let $t < T$, we will prove the result for $\mathbf{I}_t^{\rightarrow}$. The proof for the *backward interfaces* is similar. Let $\mathbf{N}_t^{\rightarrow} = \mathbf{X}_t - \mathbf{I}_t^{\rightarrow}$. Suppose there is a path from a node of $\mathbf{X}_0$ to a node of $\mathbf{X}_T$ that does not contain any node in $\mathbf{I}_t^{\rightarrow}$ in the moralized dBN. Thus, there must be an edge (u, v) such that $u \in \mathbf{N}_t^{\rightarrow}$ and $v \in \mathbf{X}_{t+1}$. If (u, v) was an arc in the non-moralized version, then u should be in $\mathbf{I}_t^{\rightarrow}$ by definition, but this leads to a contradiction. If (u, v) is an edge added during the moralization, then u and v have a common child in the non-moralized version. This child is either in $\mathbf{X}_t$ or in $\mathbf{X}_{t+1}$. If it belongs to $\mathbf{X}_{t+1}$ then u has a child in slice $t+1$ and should be in $\mathbf{I}_t^{\rightarrow}$ which leads to a contradiction. If it belongs to $\mathbf{X}_t$ then there exists an arc from the future to the past which is forbidden into a dBN. Thus the edge (u, v) can not exist and we deduce the result. □

Note that an interface is not necessarily included entirely in the same slice, but can span across several slices. Since the topology of a slice does not vary over time, the size of the backward interface, forward interface and a slice remain constant, so we can write $size(\mathbf{I}_t^{\rightarrow}) = i^{\rightarrow}$, $size(\mathbf{I}_t^{\leftarrow}) = i^{\leftarrow}$ and $size(\mathbf{X}_t) = s_{\text{slice}}$.

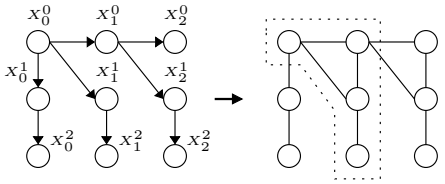


Figure 2: Triangulation of a dBN using a forward slice-by-slice elimination order where $\mathbf{I}_t^{\leftarrow} = \{X_t^0, X_t^1\}$. The subgraph surrounded in dashed line can be repeated to avoid re-triangulated the dBN if its length changes.

3 Exact inference and complexity

In this work, we have studied the *fixed interval smoothing* inference (or the *offline* inference), since all other inferences are particular cases. To perform this task, one can use two different approaches. By considering the dBN either as a stochastic process or as a Bayesian network and applying inference algorithms that are devoted for the former or the latter. In this paper, we consider the second approach and methods based on triangulation. As stated previously, the NP-hardness of finding optimal triangulation implies the use of heuristic methods. In the case of dBNs, a particular type of elimination order, called *constrained elimination order*, is distinctively interesting. A constrained elimination order is an elimination order such that we impose the elimination of a set of nodes before an other. For example, a forward (resp. backward) slice-by-slice elimination, denoted F-SS (resp. B-SS), is a constrained elimination order such that we eliminate a slice after an other from the past to the future (resp. from the future to the past) (see Figure 2). This kind of elimination orders gives a theoretical upper bound of the maximum clique size which is independent of the length of the dBN. Besides, they give good experimental results and provide a way to avoid re-triangulating the dBN for different length (Kjærulff, 1994; Bilmes and Bartels, 2003; Murphy, 2002).

In Figure 2 the backward interfaces belongs to a clique, this is a direct consequence of the following theorem:

Lemma 1 (Rose et al. (1976)). *Let $X_1, ..., X_n$ be an elimination order triangulating an undirected graph G, and let X_i and X_j two non-neighbors nodes. Then the elimination order add an edge (X_i, X_j) if and only if there exists a path with endpoints X_i and X_j such that every nodes on the path are eliminated before X_i and X_j.*

If we use a slice-by-slice elimination order then we eliminate each node in slice $t-1$ before the nodes in slice t. By definition, a backward interface is the set of nodes that have neighbors in slice $t-1$ and if there is one connected component per slice then for each pair of nodes (u, v) in the backward interface there exists a path with endpoints (u, v) where all nodes in the path are in the previous slice. Hence, by Lemma 1, (u, v) will be connected.

One can note that lemma 1 also implies that the size of the maximum clique is at least as large as the size of the backward interface. This is the main drawback of using constrained elimination order. For example, in Figure 2, the treewidth is 2 whereas the constrained triangulation gives a maximum clique of size 3. However, in practice, constrained elimination gives good results and may be superior to unconstrained elimination (Darwiche, 2001). Moreover, we have the following upper bound guarantee on the maximum clique size:

Theorem 1 (Darwiche (2001)). *Let a dBN of length T, then $m(\text{F-SS}) \leq i^{\leftarrow}.s_{slice}$, $m(\text{B-SS}) \leq i^{\rightarrow}.s_{slice}$ where $m(\sigma)$ is the size of the maximum clique obtained using the elimination order σ.*

Thus an idea to improve this result is to find a smaller interface and deduce a constrained elimination order. To illustrate this, consider Figure 3: since $\mathbf{I}_0$ is an interface we can split the dBN into two parts $\mathbf{L}_0$ and $\mathbf{R}_0$ that are respectively the nodes in the left of $\mathbf{I}_0$ and the nodes in its right. At each step, we eliminate nodes in $\mathbf{L}_i$ and slide $\mathbf{I}_i$ one time step further to get an other interface $\mathbf{I}_{i+1}$ and we repeat the process. The final step consists in eliminating nodes in $\mathbf{I}_2 \cup \mathbf{R}_2$. This elimination order denoted MINELIM is then $\mathbf{L}_0, \mathbf{L}_1 - \mathbf{L}_0, \mathbf{L}_2 - \mathbf{L}_1, \mathbf{I}_2 \cup \mathbf{R}_2$. The maximum clique dimension is then 2 which is better in comparison to the one in Figure 2.

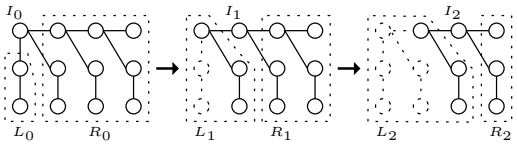


Figure 3: Finding a constrained elimination order given an interface $\mathbf{I}_0$.

This approach was followed by (Bilmes and Bartels, 2003), where the authors proposed an algorithm to find a smaller interface and an other one to build an elimination order relying on a more general model than the dBN called "GMTK-template". They show that the triangulation quality can be dramatically better than other constrained elimination.

Unfortunately, the algorithm that finds this elimination order runs in exponential time and needs to be parameterized. In the next sections, we show that finding and building such elimination order can be done in polynomial time and does not need any parameterization.

4 Minimum interface triangulation

In this section, we first prove that finding the smallest possible interface can be done in polynomial time. We then show how to build a constrained elimination order with a theoretical guarantee on the upper bound of the maximum clique size.

4.1 Finding the minimum interface

The problem of finding a minimum interface in a dBN of length T can be formally stated as:

• MINIMUM T-INTERFACE
Instance : Moralized dBN $\mathcal{G}_T^m = (\mathbf{X}_0 \cup ... \cup \mathbf{X}_T, E)$ of length T.
Solution : An interface $\mathbf{I} \subseteq \mathbf{X}_0 \cup ... \cup \mathbf{X}_T$.
Measure : $size(\mathbf{I})$

We recall the both problems MINIMUM s-t CUT and MINIMUM s-t VERTEX CUT that are used in our reduction:

• MINIMUM s-t CUT
Instance : Directed graph $G = (\{s, t\} \cup V, E)$, a weight function $w : E \rightarrow R$.

Solution : A subset $E^{'} \subseteq E$ such that every path from s to t contains an arc in $E^{'}$.

Measure : $\sum_{e \in E^{'}} w(e)$.

• MINIMUM s-t VERTEX CUT

Instance : Graph $G = (\{s,t\} \cup V, E)$, a weight function $w : V \rightarrow N$.

Solution : A subset $V^{'} \subseteq V - \{s,t\}$ such that every path from s to t contains a node in $V^{'}$.

Measure : $\sum_{v \in V^{'}} w(e)$.

We state and prove the following theorem:

Theorem 2. MINIMUM T-INTERFACE *polynomially reduces to* MINIMUM s-t CUT *and can be then solved in polynomial time.*

Proof. To prove this theorem, we use a composition of two polynomial transformations that are stated and proved in the two following lemma. We give for both lemma a sketch of proof since the proof is trivial. We refer to Figure 4 for a detailed transformation.

Lemma 2. MINIMUM T-INTERFACE *polynomially reduces to* MINIMUM s-t VERTEX CUT.

Proof. It is quite easy to see that an instance I of MINIMUM T-INTERFACE can be cast into an instance $I^{'}$ of MINIMUM s-t VERTEX CUT. Indeed, it consists essentially of adding a source and a sink to the dBN. Note that solutions of I and $I^{'}$ are of equal size. □

Lemma 3. MINIMUM s-t VERTEX CUT *polynomially reduce to* MINIMUM s-t CUT.

Proof. The transformation essentially consists of splitting each node $v \in V$ into two nodes v_i (*in-node*), v_o (*out-node*) with an arc (v_i, v_o) (*arc-node*) and a weight on this arc equal to the weight of v. An edge (u,v) is replaced by two arcs (u_o, v_i) and (v_o, u_i) with weights equal to $+\infty$. By construction, a solution of the former problem is a solution of equal size to the second problem by considering the corresponding *arc-node* and *vice versa*. □

By Lemma 2 and 3 the result follows. □

In Figure 2, when the value of T increases up to $T = 2$ the size of the minimum interface decreases. Hence, it remains open the question

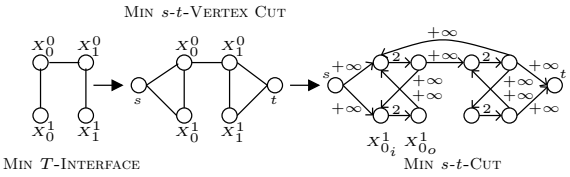


Figure 4: An example of a polynomial transformation from MINIMUM T-INTERFACE to MINIMUM s-t CUT. Variables are all binaries.

for which value of T we find the smallest interface by resolving MINIMUM T-INTERFACE. We claim that this value is less or equal to the number of nodes in a slice. Let $\mathbf{I}^{(t)}$ an optimal solution of MINIMUM T-INTERFACE with $T = t$, we first prove the following lemma:

Lemma 4. *For all $a, b \geq 0$ such that $a \geq b$,*

$$size(\mathbf{I}^{(a)}) \leq size(\mathbf{I}^{(b)})$$

Proof. For a dBN of length a, every path from a node of $\mathbf{X}_0$ to a node of $\mathbf{X}_b$ contains, by definition, a node in $\mathbf{I}^{(b)}$. Moreover, every path from a node of $\mathbf{X}_0$ to a node of $\mathbf{X}_a$ contains a node in $\mathbf{X}_b$ and then a node in $\mathbf{I}^{(b)}$. Hence, $\mathbf{I}^{(b)}$ is also an interface for the dBN of length a which implies $size(\mathbf{I}^{(a)}) \leq size(\mathbf{I}^{(b)})$. □

The following proposition give a first characterization of the optimal value for T:

Proposition 2. *Let t^* be the time such that:*

$$t^* = \min_t \{t \in N : size(\mathbf{I}^{(t)}) = size(\mathbf{I}^{(t+1)})\}$$

Then for all $t \geq 0$, we have that:

$$size(\mathbf{I}^{(t^*)}) \leq size(\mathbf{I}^{(t)})$$

Proof. By theorem 2, finding the minimum interface in a moralized dBN of length T $\mathcal{G}_T^m$ can be viewed as maximizing a flow in the directed graph $\mathcal{G}_T^{m^{'}}$ obtained from $\mathcal{G}_T^m$ by applying the polynomial transformation given in the proof. Let α_t the value of the maximum flow in the graph $\mathcal{G}_t^{m^{'}}$, by the max-flow min-cut theorem and since optimal solution of the two problems are of the same size, we can write $size(\mathbf{I}^{(t)}) = size(\mathbf{I}^{(t+1)}) = \alpha_t = \alpha_{t+1}$. Hence, in

the slice t of $\mathcal{G}^{m'}_{t+1}$ the value of the flow is maintained to the slice $t+1$. Since the topology of a dBN does not vary over time, if we could preserve a flow of value α_t from slice t to slice $t+1$, then we could preserve the flow from slice $t+1$ to slice $t+2$. Moreover, the value of the flow can not increase to a value $\alpha_{t+2} > \alpha_t$ by lemma 4. By inductively applying these arguments, we have $\alpha_t \leq \alpha_{t'}$ for all $t' > 0$ and thus $size(\mathbf{I}^{(t)}) \leq size(\mathbf{I}^{(t')})$ for all $t' > 0$. □

We can now give an upper bound of the optimal value of T:

Proposition 3. *Let a dBN with h nodes per slices, then for all $t \geq 0$, $size(\mathbf{I}^{(2h-1)}) \leq \mathbf{I}^{(t)}$.*

Proof. By lemma 4, if $t \leq 2h-1$ then $size(\mathbf{I}^{(2h-1)}) \leq size(\mathbf{I}^{(t)})$. Assume $t > 2h-1$ and $size(\mathbf{I}^{(2h-1)}) > \mathbf{I}^{(t)}$. By proposition 2, we must have $size(\mathbf{X}_0) = size(\mathbf{I}^{(0)}) > size(\mathbf{I}^{(1)}) > ... > size(\mathbf{I}^{(2h-1)}) > size(\mathbf{I}^{(t)})$. Hence, let $i \in \{0, ..., 2h-1\}$, either we reduce the size of $\mathbf{I}^{(i)}$ to get $\mathbf{I}^{(i+1)}$ by removing a node or by replacing a node with another one of lower size. We can remove at most h nodes, and replacing at most $h-1$ nodes, thus $size(\mathbf{I}^{(2h-1)}) = 0$ and $size(\mathbf{I}^{(t)}) = 0$ which leads to a contradiction. □

Using propositions 3 and 2, and the theorem 2, we deduce the `Min-Interface` algorithm (see Figure 5) that finds the smallest interface in polynomial time.

4.2 Building the elimination order

Once we have the interface $\mathbf{I}_0$ found by `Min-Interface`, the next step is to build the constrained elimination order. To do this, consider the sets $\mathbf{I}_i = \{X^k_{t+i} : X^k_t \in \mathbf{I}_0\}$ for $i = 1, ..., n$ of a dBN of length T. Each $\mathbf{I}_i$ is obtained by sliding $\mathbf{I}_0$ by i time step. As previously discussed in section 3, since $\mathbf{I}_i$ is straightforwardly an interface, we can split the dBN into two parts $\mathbf{L}_i$ and $\mathbf{R}_i$ where $\mathbf{L}_i$ is the set of nodes in the left of $\mathbf{I}_i$ and $\mathbf{R}_i$ the set of nodes in its right. We then define the following elimination order:

$$\text{MIN-ELIM} = \mathbf{L}_0, \mathbf{L}_1 - \mathbf{L}_0, ..., \mathbf{L}_n - \mathbf{L}_{n-1}, \mathbf{I}_n \cup \mathbf{R}_n$$

Require: A 2-TBN.
Ensure: A minimum interface
  prevSize $\leftarrow +\infty$
  $\mathbf{I} \leftarrow \mathbf{X}_0$.
  $T \leftarrow 1$
  while $T < 2h-1$ **do**
    prevSize $\leftarrow size(\mathbf{I})$
    $\mathcal{G}_T \leftarrow$ Unroll the 2-TBN T time steps
    $\mathcal{G}^m_T \leftarrow$ moralize($\mathcal{G}_T$)
    Solve MIN T-INTERFACE with $\mathcal{G}^m_T$ to get $\mathbf{I}$
    if $size(\mathbf{I}) =$ prevSize **then**
      return $\mathbf{I}$
    end if
    $T \leftarrow T+1$
  end while
  return $\mathbf{I}$.

Figure 5: Pseudo-code of the `Min-Interface` algorithm. Its polynomial time complexity is ensure by the $O(h)$ iterations and by the theorem 2.

Note that we do not impose any constraints in the elimination order of nodes in each $\mathbf{L}_i - \mathbf{L}_{i-1}$, $\mathbf{L}_0$ and $\mathbf{I}_n \cup \mathbf{R}_n$. In our experiments, we used the min-fill heuristic but other approaches can be considered. We now give an upper bound of the maximum clique size if we apply the elimination sequence MIN-ELIM.

Proposition 4. *For all $i > 0$, when eliminating nodes in $\mathbf{L}_i - \mathbf{L}_{i-1}$, we create a clique of size at most $size(\mathbf{I}_i).s_{slice}$ where $size(\mathbf{I}_i) \leq min(i^{\rightarrow}, i^{\leftarrow})$.*

Proof. To get $\mathbf{L}_i$ from $\mathbf{L}_{i-1}$ it suffices to add nodes X^k_t such that $X^k_{t-1} \in \mathbf{L}_{i-1}$ or $t = 0$. Then we have $\mathbf{L}_i - \mathbf{L}_{i-1} = \{X^0_{t_1}, ..., X^n_{t_n}\}$ and thus $size(\mathbf{L}_i - \mathbf{L}_{i-1}) = size(\{X^0_{t_1}, ..., X^n_{t_n}\}) = s_{\text{slice}}$. When eliminating nodes in $\mathbf{L}_i - \mathbf{L}_{i-1}$, the set of involved nodes are $(\mathbf{L}_i - \mathbf{L}_{i-1}) \cup \mathbf{I}_i$ (since $\mathbf{I}_i$ is an interface) then the size of the created clique is less or equal to $size(\mathbf{L}_i - \mathbf{L}_{i-1}).size(\mathbf{I}_i) = size(\mathbf{I}_i).s_{\text{slice}}$. Finally, by proposition 1 and the optimality of $\mathbf{I}_i$, we have $size(\mathbf{I}_i) \leq \min(i^{\rightarrow}, i^{\leftarrow})$. □

Remark. The previous proposition is a "local" improvement of theorem 1. It is "local" be-

cause when we eliminate nodes in $\mathbf{L}_0$ or in $\mathbf{I}_n \cup \mathbf{R}_n$ the created clique could be larger than $size(\mathbf{I}_i).s_{\text{slice}}$, but (1) this concerns a fix fraction of the dBN and thus the average size of the cliques in the triangulated dBN converge to $size(\mathbf{I}_i).s_{\text{slice}}$ when $T \to +\infty$ (2) we could use the B-SS (or the F-SS) elimination order to have a guarantee on the size of the maximum clique when eliminating nodes in these sets.

5 Experiments

In this section we present triangulation results on classical dBNs and random dBNs using methods based on (Ide and Cozman, 2002). To perform these tests, we used the C++ aGrUM library (`http://agrum.lip6.fr`).

		B-SS	F-SS	MIN-ELIM
Fig. 3.3 of Murphy (2002)	Mean	3.16	**3.03**	**3.03**
	Bounds	0.69–3.46	0.69–3.46	0.69–3.46
Fig. 3.2 of Murphy (2002)	Mean	**5.54**	**5.54**	**5.54**
	Bounds	5.54–5.54	5.54–5.54	5.54–5.54
Fig. 3-D of Bilmes (2003)	Mean	**1.38**	3.00	**1.38**
	Bounds	1.39–1.39	1.39–3.46	1.39–1.39
Fig. 2 of Darwiche (2001)	Mean	**2.07**	3.45	**2.07**
	Bounds	1.39–2.08	1.39–3.46	1.39–2.08
Fig. 6 of Bilmes (2003)	Mean	3.46	3.23	**3.00**
	Bounds	2.08–4.16	2.08–3.46	2.08–3.46

Figure 6: Results on real dBNs.

Figure 6 shows results for classical dBNs. Each row represents a dBN and each column corresponds to an elimination order used for triangulation. Each dBN is unrolled up to $T = 500$ time steps. We report the mean size of the cliques in the triangulated dBN and the minimum and the maximum clique. We see that our method always gives triangulated dBN that have the minimum mean size of cliques and for the last dBN it is the only one that gives the best mean size.

Figure 7 shows results for randomly generated dBN. Each dBN contains 5 ((a) and (b)), 10 ((c) and (d)) or 15 ((e) and (f)) variables per slices with random variable cardinalities chosen uniformly between 2 and 8. All dBNs are unrolled up to $T = 500$ time steps. In figure 7 (a), (c) and (e), for each generated dBN we compute the mean size of the cliques x, y_1 and y_2 in the triangulated dBN using respectively MIN-ELIM, F-SS and B-SS, then we plot the point (x, y_1) as "$\times$" and the point (x, y_2) as "$\circ$". We report in

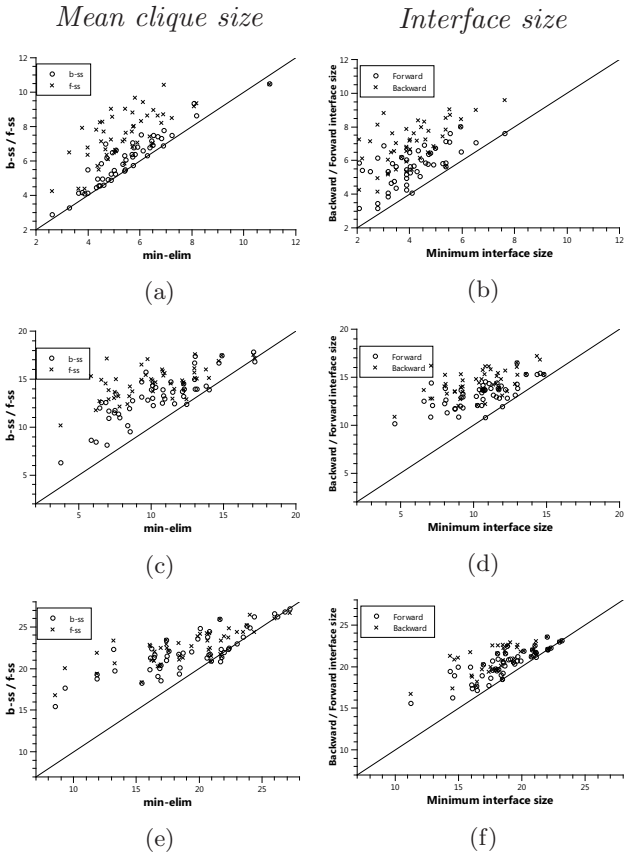


Figure 7: Results on random generated dBNs.

figure 7 (b), (d) and (f) the size of the backward, forward and minimum interface in the same way. Every points above the first diagonal shows an improvement thanks to our algorithm. As we can see, our constrained elimination order improve the triangulation quality for almost all generated dBNs.

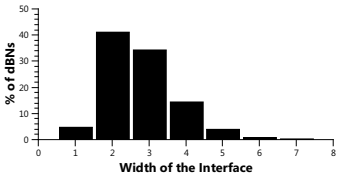


Figure 8: Minimum interfaces often span across many slices.

Figure 8 shows the percent of dBNs given the *width* of the minimum interface, *i.e.* the number of slices the interface spans across. This shows that allowing the interface to span across many slices is a useful approach.

6 Conclusion

In this paper, we studied the problem of finding good quality constrained elimination order for triangulating a dBN. We propose a polynomial algorithm to compute such order using the concept of interface in a dBN. We first show that an interface is equivalent to a cut-set in a graph, which allows us to find the minimum interface in polynomial time and then to construct a constrained elimination that have a theoretical guarantee on the maximum clique size in the triangulated graph. Experimental results show that our approach of using a polynomial time pre-treatment allows to increase the quality of the triangulation.

In future work, we plan to use this approach with approximate inference algorithms, *e.g.* loopy belief propagation, factored frontier algorithm (Murphy and Weiss, 2001), in which we could take advantage of using a small interface.

Acknowledgments

This research was supported by the ANR SKOOB project (`http://skoob.lip6.fr`).

References

Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. 1987. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284.

Cédric Baudrit, Mariette Sicard, Pierre-Henri Wuillemin, and Nathalie Perrot. 2009. Dynamic bayesian networks for modelling food processing: Application to the cheese ripening process. In *8th World Congress of Chemichal Engineering 09*, Montréal (Canada).

Jeff Bilmes and Chris Bartels. 2003. On Triangulating Dynamic Graphical Models. In *Uncertainty in Artificial Intelligence*, pages 47–56, Acapulco, Mexico.

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

Adnan Darwiche. 2001. Constant-space reasoning in dynamic Bayesian networks. *International Journal of Approximate Reasoning*, 26:161–178.

Thomas Dean and Keiji Kanazawa. 1990. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. 1999. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11:391–427.

Pinar Heggernes. 2006. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306(3):297–317.

Jaime S. Ide and Fabio G. Cozman. 2002. Random Generation of Bayesian Networks. In *Brazilian Symposium on Artificial Intelligence*, pages 366–375. Springer-Verlag.

Finn V. Jensen, Steffen L. Lauritzen, and Kristian G. Olesen. 1990. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quaterly*, 4:269–282.

Uffe Kjærulff. 1994. dHugin: A computational system for dynamic time-sliced Bayesian networks. *International Journal of Forecasting*, 11:89–111.

Kevin Murphy and Yair Weiss. 2001. The Factored Frontier Algorithm for Approximate Inference in DBNs. In *Uncertainty in Artificial Intelligence*, pages 378–385, Seattle, WA.

Kevin Murphy. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Donald J. Rose, Endre Tarjan, and Robert George S. Lueker. 1976. Algorithmic Aspects of Vertex Elimination on Graphs. *SIAM Journal on Computing*, 5(2):266–283.

Philippe Weber. 2002. Dynamic bayesian networks model to estimate process availability. In *8th International Conference Quality, Reliability, Maintenance*, Sinaia (Romania).

Geoffrey Zweig. 1996. A Forward-Backward Algorithm for Inference in Bayesian Networks and An Empirical Comparison with HMMs. Master's thesis, University of California.

Learning causal network structure from multiple (in)dependence models

Tom Claassen
Radboud University, Nijmegen
tomc@cs.ru.nl

Tom Heskes
Radboud University, Nijmegen
tomh@cs.ru.nl

Abstract

We tackle the problem of how to use information from multiple (in)dependence models, representing results from different experiments, including background knowledge, in causal discovery. We introduce the framework of a causal system in an external context to derive a connection between strict conditional independencies and causal relations between variables. Constraint-based causal discovery is shown to be decomposable into a candidate pair identification and a subsequent elimination step that can be applied separately from different models. The result is the first principled, provably sound method that is able to infer valid causal relations from different experiments in the large sample limit. We present a possible implementation that shows what results can be achieved and how it might be extended to other application areas.

1 Introduction

Discovering causal relations from observational data is an important, ubiquitous problem in science. In many application areas there is data available from many different but related experiments. Results obtained from one data set are often used to either corroborate or challenge results from another. Yet how to reconcile apparently contradictory information from multiple sources, including background knowledge, into a single, more informative model remains a long-standing open problem.

Constraint-based methods like the FCI-algorithm (Spirtes et al., 2000) are provably correct in the large sample limit, even in the presence of latent variables; the same holds for Bayesian methods like the greedy search algorithm GES (Chickering, 2002) (with additional post-processing steps to handle hidden confounders). Both are defined in terms of modeling a single data set and have no principled means to relate to results from other sources in the process. Recent developments, like the ION-algorithm by Tillman et al. (2008), show that it is possible to integrate multiple, partially overlapping data sets, provided they originate from *identical* experiments. These are still essentially single model learners as they assume there is one underlying structure that can account for *all* observed dependencies in the different models. In practice there are often inconsistencies between data sets, precisely because the experimental circumstances were not identical. The way out is to distinguish between causal dependencies internal to the system under investigation and merely contextual dependencies.

In section 4 we show that causal discovery can be decomposed into two separate steps: a conditional independency to identify a pair of possible causal relations (one of which is true), and then a conditional dependency to eliminate one of the candidates, leaving the other. The two steps are independent and rely only on the observed (in)dependencies between a subset of variables. As a result conclusions remain valid, even when taken from different models.

2 Graphical model preliminaries

First a few familiar notions from graphical model theory used throughout the article.

A *directed graph* $\mathcal{G}$ is a pair $\langle \mathbf{V}, \mathbf{E} \rangle$, where $\mathbf{V}$ is a set of vertices or nodes and $\mathbf{E}$ is a set of edges

Pp. 81–89 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

between pairs of nodes. Edges are represented by arrows $X \to Y$, where node X is the *parent* of Y and Y is a *child* of X. Two vertices are *adjacent* in $\mathcal{G}$ if there is an edge between them. A *path* $\pi = \langle V_0, \ldots, V_n \rangle$ between V_0 and V_n in $\mathcal{G}$ is a sequence of distinct vertices such that for $0 \leq i \leq n-1$, V_i and V_{i+1} are adjacent in $\mathcal{G}$. A *directed path* is a path that is traversed entirely in the direction of the arrows. A *directed acyclic graph* (DAG) is a directed graph that does not contain a directed path from any node to itself. A vertex X is an *ancestor* of Y (and Y is a *descendant* of X) if there is a directed path from X to Y in $\mathcal{G}$ or if $X = Y$. A vertex Z is a *collider* on a path $\pi = \langle \ldots, X, Z, Y, \ldots \rangle$ if it contains the subpath $X \to Z \leftarrow Y$, otherwise it is a *noncollider*. A *trek* is a path that does not contain any collider.

For disjoint sets of vertices $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ in a DAG $\mathcal{G}$, $\mathbf{X}$ is *d-connected* to $\mathbf{Y}$ conditional on $\mathbf{Z}$ (possibly empty), iff there exists an unblocked path $\pi = \langle X, \ldots, Y \rangle$ between some $X \in \mathbf{X}$ and some $Y \in \mathbf{Y}$, i.e. such that every collider on π is an ancestor of some $Z \in \mathbf{Z}$ and every noncollider on π is not in $\mathbf{Z}$. If not, then all such paths are blocked, and $\mathbf{X}$ is said to be *d-separated* from $\mathbf{Y}$ given $\mathbf{Z}$. Note that in a DAG $\mathcal{G}$, an unblocked path π between two vertices X and Y cannot be blocked by conditioning on a node Z that is not on the path, and that a blocked path can only be unblocked by conditioning on (descendants of) all colliders on the path; see (Pearl, 2000; Spirtes et al., 2000) for more details.

Let p be a probability distribution over a set of variables $\mathbf{V}$, and let $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ denote three disjoint subsets of $\mathbf{V}$, then an *(in)dependence model* is a set of (in)dependence statements that hold in p of the form '$\mathbf{X}$ is independent of $\mathbf{Y}$ given $\mathbf{Z}$', denoted $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$, and/or '$\mathbf{X}$ is dependent of $\mathbf{Y}$ given $\mathbf{Z}$', denoted $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$, with set $\mathbf{Z}$ possible empty. (In)dependence models are often compactly and intuitively represented in the form of a graphical model (directed, undirected or other), in combination with a criterion to link the structure of the graph to the implied (in)dependencies, similar to the d-separation for DAGs. We will pose no restrictions on shape or type of the (in)dependence models considered in this article, other than that they are internally consistent.

3 Modeling the system

This section introduces the framework of a causal system in an external context to model experiments, as well as a number of assumptions adopted throughout the rest of this article.

3.1 Causal DAG

A causal DAG is a graphical model in the form of a DAG where the arrows represent direct causal interactions between variables in a system. A prime characteristic of a causal structure is the so-called *Manipulation Principle* (Spirtes et al., 2000), which boils down to the fact that changing/manipulating a variable will affect all and only its descendants in the causal DAG. In this article we will not concern ourselves with the interpretation of causality any further; for that the reader is referred to (Cartwright, 2004; Williamson, 2005). Instead, we simply assume that the systems we consider can be represented by some underlying causal DAG over a great many observed and unobserved nodes. In a causal DAG $\mathcal{G}_C$ there is a *causal relation* from variable X to variable Y iff there is a directed path π from X to Y in $\mathcal{G}_C$, otherwise it is a *noncausal relation*. A direct link $X \Rightarrow Y$ in the graph $\mathcal{G}_C$ means that there is a causal path from X to Y that is not mediated by any other node in $\mathcal{G}_C$.

The ubiquitous **causal Markov condition** links the structure of a causal graph to its probabilistic concomitant, (Pearl, 2000): two variables X and Y in a causal DAG $\mathcal{G}_C$ are dependent given a set of nodes $\mathbf{Z}$, iff they are connected by a path π in $\mathcal{G}_C$ that is unblocked given $\mathbf{Z}$. An immediate consequence is that there is a dependence $X \not\perp\!\!\!\perp Y$ iff there is a trek between X and Y in the causal DAG.

Another common assumption which we will adopt throughout the article is the **causal faithfulness condition** which implies that all and only the conditional independence relations entailed by the causal Markov condition applied to the true causal DAG will hold in the joint probability distribution over the variables in $\mathcal{G}_C$.

For an in-depth discussion of the justification of and connection between these assumptions in causal inference, see (Pearl, 2000; Spirtes et al., 2000; Zhang and Spirtes, 2008).

3.2 Experimental context

Random variation in a system (a.k.a. 'error terms' in a structural equation model (SEM)), corresponds to the impact of unknown external variables (Pearl, 2000). Some external factors may be actively controlled, as for example in clinical trials, or passively observed as the natural embedding of a system in its environment. We refer to both observational and controlled studies as *experiments*. If there are external factors that affect two or more variables in a system simultaneously, then this can lead to an observed dependency that is not part of the system (a.k.a. 'correlated errors' in SEMs). Both can be represented by modeling this external environment explicitly as a set of unknown, hypothetical context nodes that causally affect the system under scrutiny. We introduce:

Definition 1. The *external context* of a causal DAG $\mathcal{G}_C$, denoted $\mathcal{G}_E$, is an additional set of mutually independent nodes $\mathbf{U}$ in combination with links from every $U \in \mathbf{U}$ to one or more nodes in $\mathcal{G}_C$.

The total causal structure of an experiment on a causal system $\mathcal{G}_C$ in external context $\mathcal{G}_E$ is then denoted by $\mathcal{G}_T = \{\mathcal{G}_E + \mathcal{G}_C\}$. The context only introduces arrows from nodes in $\mathcal{G}_E$ to $\mathcal{G}_C$ which can never result in a cycle if there was not one in $\mathcal{G}_C$ already (there are no links between nodes in $\mathcal{G}_E$). Therefore, the structure of an experiment $\mathcal{G}_T$ is also a causal DAG. In this paradigm different experiments become variations in context of an *invariant* causal system.

Figure 1 depicts a causal system in two different contexts (double lined arrows indicate direct causal relations; dashed circles represent unobserved variables). The experiment on the right hand side will result in an observed dependency between variables A and B, whereas the one on the left will not.

Here we only focus on the (in)dependence relations $\mathcal{I}(\mathbf{V} \subset \mathcal{G}_C)$ that exist in the joint probability distribution $P(\mathbf{V})$ over the observed subset of variables for a given causal experiment $\{\mathcal{G}_E + \mathcal{G}_C\}$. With this we can state the goal of causal discovery from multiple models as: "Given experiments with unknown total causal structures $\mathcal{G}_T = \{\mathcal{G}_E + \mathcal{G}_C\}$, $\mathcal{G}'_T = \{\mathcal{G}'_E + \mathcal{G}_C\}$, etc., and corresponding (in)dependence models $\mathcal{I}(\mathbf{V} \subset \mathcal{G}_C)$, $\mathcal{I}'(\mathbf{V}' \subset \mathcal{G}_C)$, etc., which variables are connected by a directed path in $\mathcal{G}_C$?". We assume that in each experiment the large sample limit distributions are known and have been used to obtain categorical statements about probabilistic (in)dependencies between sets of nodes. As stated, we will also always assume that the causal Markov and causal faithfulness condition are satisfied.

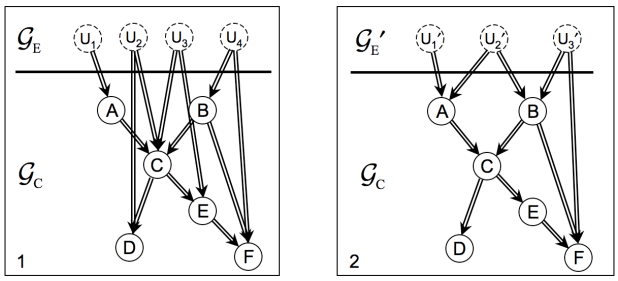


Figure 1: A causal system $\mathcal{G}_C$ in different experiments

4 Conditional (in)dependence in causal systems

Given the problem statement above, we need a way to combine (in)dependence statements from different models in order to identify causal relations in the underlying causal structure $\mathcal{G}_C$ that is assumed to be at the heart of all of them. Methods like FCI and GES tackle this reconstruction problem in terms of properties that are optimal or minimal w.r.t. a model for a given experiment, but this gives no means to relate results from different models. Another approach, taken in the ION algorithm, is to use ancestral graph theory (Richardson and Spirtes, 2002) to establish what probabilistic (in)dependencies will be observed in a causal experiment for different subsets of observed variables, and then use this to find relations that must be shared by all. But this still does not allow to combine

results from *different* experiments, like in fig. 1.

A way out of this predicament comes courtesy of a remarkable fact that so far (to the best of our knowledge) has escaped detection in causal research: there is a fundamental connection between causality and a certain type of conditional independence, that applies regardless of the encompassing model. This connection will enable us to bring together results from arbitrary experiments in a method for causal discovery from multiple (in)dependence models (section 6). To exclude irrelevant independencies we first introduce the following notion:

Definition 2. Two nodes X and Y are *strictly conditionally (in)dependent* given a set of nodes $\mathbf{Z}$, iff X is conditionally (in)dependent of Y given a *minimal* set of nodes $\mathbf{Z}$.

We denote a strict (in)dependence statement by placing it in square brackets. The *minimal* in the definition implies that the relation does not hold for any proper subset of the (possibly empty) set $\mathbf{Z}$, e.g. a strict conditional independence $[X \perp\!\!\!\perp Y \,|\, \mathbf{Z}]$ implies both $X \perp\!\!\!\perp Y \,|\, \mathbf{Z}$ and $\forall \mathbf{Z}' \subsetneq \mathbf{Z} : X \not\perp\!\!\!\perp Y \,|\, \mathbf{Z}'$. It aims to capture the notion that it is really the entire set $\mathbf{Z}$ that makes X and Y independent. The relevance of this notion lies in the fact that, in a causal system, certain causal relations between three variables X, Y and Z can *never* result in an observed strict conditional independence $[X \perp\!\!\!\perp Y \,|\, Z]$, no matter what the context is.

Example 1. For the causal system $\mathcal{G}_C$ in fig.2a (two variables $X \Rightarrow Z$ with no causal links to or from a variable Y), there is no context $\mathcal{G}_E$ that can result in $[X \perp\!\!\!\perp Y \,|\, Z]$: if there are no directed paths from Z to X and Y then $X \not\perp\!\!\!\perp Y$ implies that X and Y are d-connected by directed paths $\langle U, \ldots, X \rangle$ and $\langle U, \ldots, Y \rangle$ that do not contain Z. But then conditioning on Z cannot block these paths, ergo not $X \perp\!\!\!\perp Y \,|\, Z$. This does not apply to causal system in fig.2b: for the indicated context $\mathcal{G}_E$ the strict conditional independence relation $[X \perp\!\!\!\perp Y \,|\, Z]$ will be observed.

A quick survey shows that all causal structures over three nodes that *can* lead to an observed $[X \perp\!\!\!\perp Y \,|\, Z]$ have a direct causal link from Z to X and/or Y.

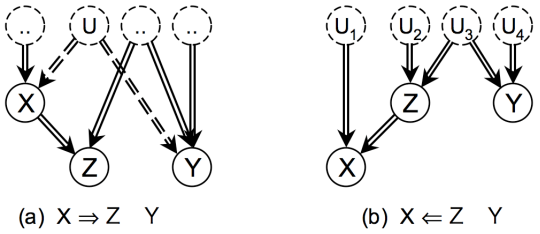


Figure 2: Causal systems $\mathcal{G}_C$ that: (a) cannot, and (b) depending on the context $\mathcal{G}_E$ *can* lead to an observed strict conditional independence relation $[X \perp\!\!\!\perp Y \,|\, Z]$.

We can generalize this result to sets of nodes:

Theorem 1. *In an experiment with causal structure $\mathcal{G}_T = \{\mathcal{G}_E + \mathcal{G}_C\}$, a strict conditional independence $[X \perp\!\!\!\perp Y \,|\, \mathbf{Z}]$ implies causal links $Z \Rightarrow X$ and/or $Z \Rightarrow Y$ from every $Z \in \mathbf{Z}$ to X and/or Y in $\mathcal{G}_C$.*

Proof. We construct a directed path for an arbitrary $Z_1 \in \mathbf{Z}$ to either X or Y. Z_1 must be a noncollider on some path π_1 connecting X and Y given all the other nodes $\mathbf{Z}_{\setminus Z_1}$. Follow π_1 in the direction of the arrows (choose either branch if Z_1 has two outgoing arrows along π_1) until either X or Y or a collider that is an ancestor of one of the remaining nodes in $\mathbf{Z}_{\setminus Z_1}$ is encountered. If X or Y is found first then a directed path has been found and we are done. If not then we can go on from the collider along π_1 to its descendant node $Z_2 \in \mathbf{Z}_{\setminus Z_1}$. This node in turn must be a noncollider on some other path π_2 that d-connects X and Y given all nodes $\mathbf{Z}_{\setminus Z_2}$. Again this path can be followed in the direction of the arrows until either X or Y or a collider that is ancestor of one of the nodes in $\mathbf{Z}_{\setminus \{Z_1, Z_2\}}$ is encountered. (This cannot be one of the previous nodes since that would imply the existence of a directed path.) We can continue, and as long as neither X nor Y is reached we will find new nodes from $\mathbf{Z}$ until all have been encountered. At that point the final node will lie on a trek connecting X and Y that can no longer be blocked by any other node in $\mathbf{Z}$, and therefore will have a directed path to X or Y. By construction that means there is also a directed path from Z_1 to either X or Y in $\mathcal{G}_C$, which implies a causal relation $Z_1 \Rightarrow X$ and/or $Z_1 \Rightarrow Y$. □

This theorem recognizes conditional independence as the 'local signature' of causality. It is not difficult to see that for a single Z the causal link to X or Y is also *unconfounded* (no hidden common parent). This plays an important role in calculating the magnitude of causal effects, e.g. via the front-door criterion (Pearl, 2000).

A similar result exists for conditional dependence and noncausal relations, something we already knew for v-structures (unshielded colliders $X \rightarrow Z \leftarrow Y$) from (Spirtes et al., 2000), although not in the general form given here:

Theorem 2. *Let X, Y, $\mathbf{Z}$ and $\mathbf{W}$ be disjoint (sets of) nodes in an experiment with causal structure $\mathcal{G}_T = \{\mathcal{G}_E + \mathcal{G}_C\}$. If there is a conditional independence $X \perp\!\!\!\perp Y \,|\, \mathbf{W}$ and a minimal set $\mathbf{Z}$ such that $X \not\perp\!\!\!\perp Y \,|\, \{\mathbf{W} \cup \mathbf{Z}\}$, then there are no causal links $Z \Rightarrow X$, $Z \Rightarrow Y$, and/or $Z \Rightarrow W$ from any $Z \in \mathbf{Z}$ to any X, Y and/or $W \in \mathbf{W}$ in $\mathcal{G}_C$.*

Proof. We show it holds for arbitrary $Z \in \mathbf{Z}$. In short: Z must be a (descendant of a) collider on a path connecting X and Y (otherwise it would not be needed to unblock the path); any directed path from Z to a W implies that conditioning on Z is not needed when already conditioning on W. No directed paths from Z to $\mathbf{W}$ implies that if there existed a directed path from Z to X or Y then it cannot be blocked by any W; neither can it be blocked by any $\mathbf{Z}_{\setminus Z}$ (otherwise $\mathbf{Z}$ is not minimal). But then such a path would make Z a noncollider on an unblocked path between X and Y given $\mathbf{Z}_{\setminus Z}$, contradicting minimality. □

With the addition of $\mathbf{W}$ the theorem also applies to unshielded colliders where X and Y are not independent. We need one more result that is particularly useful to eliminate direct links between variables in a causal model:

Theorem 3. *In an experiment with causal structure $\mathcal{G}_T = \{\mathcal{G}_E + \mathcal{G}_C\}$, every conditional independence $X \perp\!\!\!\perp Y \,|\, \mathbf{Z}$ implies the absence of causal paths $X \Rightarrow Y$ or $X \Leftarrow Y$ in $\mathcal{G}_C$ between X and Y that are not mediated by nodes in $\mathbf{Z}$.*

Proof. If there did exist causal paths between X and Y not mediated by $\mathbf{Z}$ then conditioning on $\mathbf{Z}$ would not block all directed paths (let alone treks) between X and Y, so then $X \not\perp\!\!\!\perp Y \,|\, \mathbf{Z}$. □

5 Identifying causal relations

Theorem 1 and 2 together show that causal discovery can be decomposed into two *separate* steps: having a means of identifying a pair of links that harbors a causal relation as well as a means of eliminating a causal relation as the origin of an observed link, the obvious consequence is that this allows the positive identification of a definite causal relation.

Corollary 1. *In an experiment $\mathcal{G}_T = \{\mathcal{G}_E + \mathcal{G}_C\}$, if there exists a strict conditional independence $[X \perp\!\!\!\perp Y \,|\, \mathbf{Z}]$, then if there also exists a conditional independence $X \perp\!\!\!\perp V \,|\, \mathbf{W}$ and $\mathbf{Z}$ is a minimal set such that $X \not\perp\!\!\!\perp V \,|\, \{\mathbf{W} \cup \mathbf{Z}\}$, then there are causal links $Z \Rightarrow Y$ from every $Z \in \mathbf{Z}$ to Y in $\mathcal{G}_C$.*

Proof. By theorem 1 $[X \perp\!\!\!\perp Y \,|\, \mathbf{Z}]$ implies causal links from every $Z \in \mathbf{Z}$ to X and/or Y. The second condition, $X \perp\!\!\!\perp V \,|\, \mathbf{W}$ with $\mathbf{Z}$ minimal such that $X \not\perp\!\!\!\perp V \,|\, \{\mathbf{W} \cup \mathbf{Z}\}$, applies to theorem 2 and implies that there are no causal links from any $Z \in \mathbf{Z}$ to X. With all links from $\mathbf{Z}$ to X eliminated, the only remaining option is causal links $Z \Rightarrow Y$ from every $Z \in \mathbf{Z}$ to Y. □

To illustrate how these rules can be applied to infer causal links directly from observed (in)dependence relations, we look at two independence models (represented in figure 3 as CPAGs, see Appendix A), that are known, e.g. from the FCI-algorithm (Spirtes et al., 2000), to contain a definite causal link, and show how this also follows as a straightforward application of theorems 1 and 2.

Example 2. The aptly named *Y-structure* in the l.h.s. of fig. 3 plays an important role in causal discovery: every such substructure in a minimal independence model derived by the FCI-algorithm allows the identification of causal link $Z \Rightarrow Y$, i.e. a directed path from Z to Y is present in *all* possible causal DAGs corresponding to the observed distribution over the variables. Mani et al. (2006) investigated marginal Y-structures *embedded* in data sets. It

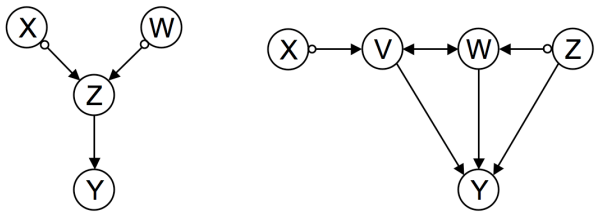


Figure 3: Two independence models in the form of a CPAG: the 'Y-structure' (left) and a discriminating path (right), both with a detectable causal link $Z \Rightarrow Y$ (arrow in CPAG); see examples for detailed description.

was shown that for any DAG, in the large sample limit, a consistent Bayesian scoring function (Heckerman et al., 1999) will assign a higher score to a structure with a direct link $Z \rightarrow Y$, when marginalizing over the variables, than to any structure without. These results are easily understood in terms of our theorems: any (embedded) Y-structure satisfies the relations $[X \perp\!\!\!\perp Y \mid Z]$ and $[X \not\perp\!\!\!\perp W \mid Z]$. By theorem 1, the first implies $Z \Rightarrow X$ or $Z \Rightarrow Y$, the second eliminates $Z \not\Rightarrow X$ by theorem 2, leaving $Z \Rightarrow Y$.

As another example, we look at the following important, but somewhat awkward, construct in causal inference: in a graph $\mathcal{G}$, a path $\pi = \langle X, \ldots, W, Z, Y \rangle$ is a *discriminating path* for Z if X is not adjacent to Y and all nodes between X and Z are colliders on π and parents of Y.

Example 3. The path $\pi = \langle X, V, W, Z, Y \rangle$ in the r.h.s. of figure 3 is a discriminating path for Z. The relevance of such a path for causal discovery lies in the fact that if $Z \rightarrow Y$ is present in the graph $\mathcal{G}$, then it is present in all members of the equivalence class of $\mathcal{G}$, and hence it corresponds to a definite causal link (Spirtes et al., 2000; Zhang, 2008). The causal implication of this discriminating path can also be understood in terms of the previous rules: by definition of π it follows that X and Y are strict conditionally independent given some set $\mathbf{Z}$ (otherwise they would be adjacent in $\mathcal{G}$). If there is a link $Z \rightarrow Y$, then Z (and all other nodes between X and Z on π) is necessarily part of any $\mathbf{Z}$ that will d-separate X and Y. Therefore, figure 3 implies $[X \perp\!\!\!\perp Y \mid Z \cup \{V, W\}]$ and $X \not\perp\!\!\!\perp Z \mid \varnothing$, which by theorems 1 and 3 implies $Z \Rightarrow Y$.

6 Causal relations from multiple models

As all three theorems (rules) in section 4 hold separately for experiments $\mathcal{G}_T$ *irrespective* of the context $\mathcal{G}_E$, it means that (non)causal results obtained in one experiment should also apply to another, provided the causal system $\mathcal{G}_C$ remains invariant. In that case, an algorithm implementing these rules should be able to construct a single, overall model of the causal relations that is more informative than any of the (in)dependence models separately.

For that we note that all noncausal information ('X does not cause Y') from rules (2) and (3) derives from single models in isolation, and so can be processed first and collected in a matrix of (non)causal relations found. Subsequent causal relations identified via rule (1) also imply reverse noncausal information, which in turn can lead to new causal relations. This suggests a repeated loop until no new information can be found. As input for the algorithm we use CPAGs (see Appendix A) as a concise and intuitive graphical representation of all invariant (in)dependence features in an observed distribution, e.g. as learned by the extended FCI-algorithm (Zhang, 2008). To convey all uncovered information about the underlying causal structure $\mathcal{G}_C$ we choose a *causal PAG* $\mathcal{G}$ as the output model: similar in form and interpretation to a CPAG, where a missing edge between variables corresponds to the absence of a direct causal path, every detected direct noncausal link $X \not\Rightarrow Y$ has an arrowhead at X in $\mathcal{G}$, every detected direct causal link $X \Rightarrow Z$ has a tail mark at X in $\mathcal{G}$, and circle marks represent unknown, possibly causal relations. A straightforward implementation is provided in algorithm 1.

To illustrate the algorithm, consider the CPAG models corresponding to two experiments on the l.h.s. of figure 4. Despite the different, even apparently contradictory, observed (in)dependence relations, the combined causal model on the r.h.s. is readily derived.
Starting from the fully connected graph, in the first loop over the models, rule (3) in line

Input : set of CPAGs $\mathcal{P}_i \in \mathbf{P}$
Output : causal graph $\mathcal{G}$
1: $\mathcal{G} \leftarrow$ fully connected graph with circle marks
2: $\mathbf{M}_C \leftarrow \mathbf{0}$ ▷ *empty set of (non-)causal relations*
3: **for all** $\mathcal{P}_i \in \mathbf{P}$ **do**
4: **for all** $(X, Y, Z) \in \mathcal{P}_i$, with no edge $X - Y$ **do**
5: $\mathbf{M}_C \leftarrow X \not\leftarrow Y$, $Y \not\leftarrow X$ if $X \perp\!\!\!\perp Y \mid \varnothing$ ▷ *Rule (3)*
6: **for all** $\mathbf{W} \in \{\mathcal{P}_i \setminus X, Y, Z\}$ **do**
7: **if** $X \perp\!\!\!\perp Y \mid \mathbf{W}$ **then**
8: $\mathcal{G} \leftarrow$ eliminate edge $X \multimap Y$ ▷ *Rule (3)*
9: **if** $X \not\perp\!\!\!\perp Y \mid \{\mathbf{W} \cup Z\}$ **then**
10: $\mathbf{M}_C \leftarrow Z \not\leftarrow \{X, Y, \mathbf{W}\}$ ▷ *Rule (2)*
11: **end if**
12: **end if**
13: **end for**
14: **end for**
15: **end for**
16: $\mathcal{G} \leftarrow$ noncausal info in $\mathbf{M}_C$ ▷ *circles to arrowheads*
17: **repeat**
18: **for all** $\mathcal{P}_i \in \mathbf{P}$ **do**
19: **for all** $(X, Y) \in \mathcal{P}_i$, with no edge $X - Y$ **do**
20: **for all** $\mathbf{Z} \in \{\mathcal{P}_i \setminus X, Y\}$ **do**
21: **if** $[X \perp\!\!\!\perp Y \mid \mathbf{Z}]$ **and** $X \not\leftarrow Z \in \mathbf{M}_C$ **then**
22: $\mathbf{M}_C \leftarrow Z \Rightarrow Y$ **and** $Y \not\leftarrow Z$ ▷ *Rule (1)*
23: **end if**
24: **end for**
25: **end for**
26: **end for**
27: $\mathcal{G} \leftarrow$ (non)causal info in $\mathbf{M}_C$ ▷ *tails/arrowheads*
28: **until** no more new noncausal information found

Algorithm 1: Causal structure inference algorithm

8 eliminates all links except $A - C$, $B - C$, $B - F$, $C - D$, $C - E$ and $E - F$ (missing edges in input model). In the same loop, model 1 has $[A \not\perp\!\!\!\perp B \mid \{C/D/E/F\}]$ which by rule (3) in line 5 implies $A \not\leftarrow B$ and $B \not\leftarrow A$, and from which rule (2) in line 10 derives noncausal links $\{C/D/E/F\} \not\leftarrow \{A, B\}$ (for empty $\mathbf{W}$ in theorem 2) . In the subsequent repeated loop, lines 17-28, model 1 has $[A \perp\!\!\!\perp F \mid \{B, C\}]$ which by rule (1) in line 22 with the earlier $B \not\leftarrow A$, implies $B \Rightarrow F$. Similarly, $[C \perp\!\!\!\perp F \mid \{B, E\}]$ allows the conclusion $E \Rightarrow F$. Next, model 2 has $[A \perp\!\!\!\perp D \mid C]$ which, together with $C \not\leftarrow A$ implies $C \Rightarrow D$. Finally, from $[A \perp\!\!\!\perp E \mid C]$ follows $C \Rightarrow E$. After that the algorithm terminates at line 28 with the causal CPAG on the r.h.s. as the final output. (Figure 1 shows two contexts that can account for the observed dependencies in figure 4).

To the best of our knowledge, this is the first algorithm ever to perform such a derivation. The input in the form of CPAGs is convenient, but not essential: any (in)dependence

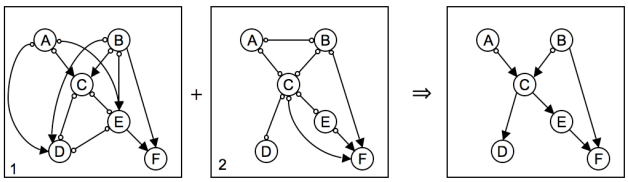


Figure 4: Independence models (in CPAG form) for two experiments, one resulting causal model (cf. fig.1).

model can be used with only minor alterations to the implementation. We could even directly incorporate (non)causal information from background knowledge in the first loop. In the current form the example derivation is almost instantaneous, but soon becomes unfeasible for larger networks. Also the set of observed variables can differ between input models, but with little overlap causal information may be lost if it cannot be transferred to the output graph when other information has eliminated that particular direct link. Nevertheless, all identified (non)causal relations remain valid. These problems can be addressed and significant improvements can be made, but that requires additional results and explication and will be postponed to another article.

7 Discussion

We have shown the first principled method to use information from *different* (in)dependence models in causal discovery. It is based on the discovery of a fundamental property that identifies (strict) conditional independence as the local signature of causality. All (non)causal relations uncovered this way are sound, provided the input models are valid. The number and individual size and origin of the input models are irrelevant and could include different experiments, specific background knowledge or hypothetical information. An exciting possibility is to use this approach in combination with recent developments that employ other properties of the distribution, e.g. non-Gaussianity (Shimizu et al., 2006) or nonlinear features (Hoyer et al., 2009), to detect causal relations.

The proposed algorithm is sound and works well on small models ($\lesssim 10$ nodes) with a rea-

sonable degree of overlap. In order to apply the method to larger, more realistic models with less overlap, further research should concentrate on the computational complexity of the search for (new) strict conditional independencies and ways to handle indirect causal information. If the input models become less reliable, for example when derived from real data sets where the large sample limit no longer applies, incorrect or inconsistent causal conclusions may occur. In that case, results might be generalized to quantities like 'the probability of a causal relation' based on the strength and reliability of the required conditional (in)dependencies in the available data.

Acknowledgments

This research was supported by VICI grant 639.023.604 from the Netherlands Organization for Scientific Research (NWO).

Appendix A. CPAGs

For a causal DAG the distribution over a subset of observed variables may not be faithfully representable by a DAG. A *complete partial ancestral graph* (CPAG) $\mathcal{P}$ represents the Markov equivalence class $[\mathcal{G}]$ of a DAG $\mathcal{G}$ when latent variables may be present (Zhang, 2008). It is a graph with either a tail '$-$' (signifying ancestorship), arrowhead '$\blacktriangleright$' (signifying non-ancestorship) or circle mark '$\circ$' at each end of an edge. There is a tail or arrowhead on an edge in $\mathcal{P}$ iff it is invariant in $[\mathcal{G}]$, otherwise it has a circle mark. Bi-directed edges $\blacktriangleleft\!\!\blacktriangleright$ in a CPAG indicate the presence of a latent common cause; arcs $\longrightarrow\!\!\blacktriangleright$ indicate a causal relation. The CPAG is unique and maximally informative for $[\mathcal{G}]$. An intuitive property of CPAGs is that two nodes X and Y are not connected by an edge iff there is some set $\mathbf{Z}$ such that $X \perp\!\!\!\perp Y \mid \mathbf{Z}$; see (Richardson and Spirtes, 2002; Zhang, 2008) for more information on how to read (in)dependencies directly from a CPAG using the m-separation criterion.

References

N. Cartwright. 2004. Causation: one word, many things. *Philosophy of Science*, (71):805–819.

D. Chickering. 2002. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(3):507–554.

D. Heckerman, C. Meek, and G. Cooper. 1999. A Bayesian approach to causal discovery. In *Computation, Causation, and Discovery*, pages 141–166.

P. Hoyer, D. Janzing, J. Mooij, J. Peters, and B. Schölkopf. 2009. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems 21 (NIPS*2008)*, pages 689–696.

S. Mani, G. Cooper, and P. Spirtes. 2006. A theoretical study of Y structures for causal discovery. In *Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence*, pages 314–323.

J. Pearl. 2000. *Causality: models, reasoning and inference*. Cambridge University Press.

T. Richardson and P. Spirtes. 2002. Ancestral graph Markov models. *Ann. Stat.*, 30(4):962–1030.

S. Shimizu, P. Hoyer, A. Hyvärinen, and A. Kerminen. 2006. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7:2003–2030.

P. Spirtes, C. Glymour, and R. Scheines. 2000. *Causation, Prediction, and Search*. The MIT Press, Cambridge, Massachusetts, 2nd edition.

R. Tillman, D. Danks, and C. Glymour. 2008. Integrating locally learned causal structures with overlapping variables. In *Advances in Neural Information Processing Systems, 21*.

J. Williamson. 2005. *Bayesian nets and causality: philosophical and computational foundations*. Oxford University Press, Oxford.

J. Zhang and P. Spirtes. 2008. Detection of unfaithfulness and robust causal inference. *Minds and Machines*, 2(18):239–271.

J. Zhang. 2008. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873 – 1896.

An Influence Diagram Model for Detecting Credit Card Fraud

Barry R. Cobb
Virginia Military Institute
cobbbr@vmi.edu

Abstract

A hybrid influence diagram is a compact graphical and numerical representation of a decision problem under uncertainty that includes both discrete and continuous chance variables. These models can be used by businesses to detect online credit card transactions that may be fraudulent. By creating decision rules based on merchandise value and additional address and product characteristics, the influence diagram model can be used to develop policies that help businesses decide when to investigate an order's legitimacy. The influence diagram establishes guidelines that minimize the sum of the costs of lost merchandise and order investigation.

1 Introduction

A major credit card issuer—Visa—encourages businesses to prevent "card-not-present fraud" by developing "...in-house fraud detection programs, such as guidelines for staff on how to spot and report suspected fraudulent transactions" ("Credit card fraud," 2008) and lists the following common characteristics of falsified credit card orders: 1) first-time orders, 2) larger than normal orders, 3) orders consisting of several of the same item, 4) orders shipped rush or overnight, and 5) orders shipped to a foreign address.

Fraud detection methods have been implemented using a number of quantitative techniques in the fields of data mining, statistics, and artificial intelligence. Cobb (2010) provides a survey of several of these methods. A large portion of the previous research that adapts quantitative techniques to credit card fraud detection has focused on these problems from the perspective of banks and firms that issue credit cards. However, businesses that accept credit cards for purchases—particularly in online transactions—can also benefit from the application of such models.

Some credit card fraud can be prevented prior to the customer completing a purchase. For instance, when a credit card approval for the purchase amount is requested, an order being submitted on a card that has been reported as stolen is denied. To further prevent fraud, a business can collect additional information, such as the three-digit card verification value on the back of the card. This prevents someone who has obtained a stolen credit card number, but not the actual card, from completing a fraudulent transaction. However, if a card has not been discovered as stolen, the fraud will not be detected at this point.

This paper proposes an influence diagram (ID) framework that can be employed to develop processes by which businesses can select transactions for further investigation as potentially falsified. The goal of the model is to appropriately balance the cost of shipping merchandise that will ultimately not be paid for because of a fraud-related chargeback, versus the cost of utilizing employee time and system resources to confirm and investigate potentially fraudulent orders. An ID is a probabilistic model that is a simultaneous graphical and numerical representation of a decision problem under uncertainty (Howard and Matheson, 1984). Recent innovations in IDs permit models with non-Gaussian continuous chance variables and discrete decision variables (Cobb and Shenoy, 2008). In this paper, a model that allows a business to develop an optimal decision rule for

Pp. 89–97 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

whether or not to investigate a transaction for fraud based on the observation of both discrete and continuous variables is suggested.

The remainder of this paper is organized as follows. Section 2 provides notation and definitions. Section 3 describes the ID model. Section 4 illustrates the solution of the ID model for an example fraud detection problem. Section 5 summarizes the paper. This paper is derived from a longer working paper on this topic (Cobb, 2010).

2 Notation and Definitions

This section introduces notation and definitions used throughout the paper.

2.1 Notation

Variables are denoted by capital letters in plain text, e.g., A, B, C. Sets of variables are denoted by capital letters in boldface, e.g., $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$. If A and $\mathbf{X}$ are one- and multi-dimensional variables, respectively, then a and $\mathbf{x}$ represent specific values of those variables. The state space of $\mathbf{X}$ is denoted by $\Omega_{\mathbf{X}}$.

A probability potential, ϕ, for $\mathbf{X}$ is a function $\phi : \Omega_{\mathbf{X}} \rightarrow [0, 1]$. If A is discrete, the more intuitive notation $P(A)$ may be used to represent a discrete probability distribution. A utility potential, u, for a set of variables $\mathbf{X}$ is a function $u : \Omega_{\mathbf{X}} \rightarrow \mathcal{R}$.

2.2 Mixtures of Truncated Exponentials

One difficulty associated with including continuous chance variables in IDs is that mathematical operations, such as integration, on probability density functions are difficult to perform in closed form. For the case where all chance variables are normally distributed and discrete variables do not have continuous parents, the technique of Madsen and Jensen (2005) can be applied to solve the ID.

For problems with continuous variables that are not normally distributed, the state spaces must be discretized to permit an ID solution or a mixture-of-Gaussians ID model (Poland and Shachter, 1993) can be used. Another approach is to approximate probability density functions in the ID with mixtures of truncated exponentials (MTE) potentials, which are defined as follows.

Definition 1. (MTE Potential (Moral et al., 2001)). Let S be a continuous chance variable. Given a partition $\Omega_1, \ldots, \Omega_n$ that divides Ω_S into hypercubes, an n-piece MTE potential $\phi : \Omega_S \mapsto \mathcal{R}^+$ has components

$$\phi_h(s) = a_0 + \sum_{i=1}^{m} a_i \ exp\left\{b_i \cdot s\right\}$$

for $h = 1, \ldots, n$, where $a_i, i = 0, \ldots, m$ and b_i, $i = 1, \ldots, m$ are real numbers.

MTE potentials can be used to approximate both probability distributions and utility functions. The optimization procedure outlined by Cobb et al. (2006) is used to determine the parameters (the values a_i and b_i) required to approximate probability density functions with MTE potentials.

2.3 Operations on MTE Potentials

In this paper, the operations of combination and marginalization are used to solve IDs where MTE potentials are used to represent probability density functions.

Definition 2. (**Combination.**) Combination of MTE potentials is pointwise multiplication. Let ϕ_1 and ϕ_2 be MTE potentials for $\mathbf{X}_1$ and $\mathbf{X}_2$. The combination of ϕ_1 and ϕ_2 is a new MTE potential for $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$ defined as follows

$$\phi(\mathbf{x}) = (\phi_1 \otimes \phi_2)(\mathbf{x}) = \phi_1(\mathbf{x}^{\downarrow \Omega_{\mathbf{X}_1}}) \cdot \phi_2(\mathbf{x}^{\downarrow \Omega_{\mathbf{X}_2}})$$

for all $\mathbf{x} \in \Omega_{\mathbf{X}}$.

Combination of two MTE probability densities results in an MTE probability density. Combination of an MTE probability density and an MTE utility potential results in an MTE utility potential, as does the combination of two MTE utility potentials. Note that since a discrete probability distribution is a special case of an MTE potential where $a_1, \ldots, a_m$ in each component are equal to zero, this definition of combination applies to discrete probability distributions.

Definition 3. (**Marginalization of Chance Variables.**) Marginalization of a chance variable is summation over its state space. Let ϕ be an MTE potential for $\mathbf{X} = \mathbf{X}' \cup X$. The state space of X is $\Omega_X = \{x_1, \ldots, x_n\}$. The marginal of ϕ for a set of variables $\mathbf{X}'$ is an MTE potential computed as

$$\phi^{\downarrow \mathbf{X}'}(\mathbf{x}') = \phi^{-X}(\mathbf{x}') = \sum_{i=1}^{n} \phi(X = x_i, \mathbf{x}') \quad (1)$$

for all $\mathbf{x}' \in \Omega_{\mathbf{X}'}$. If the variable X is a continuous chance variable, the summation in Eq. (1) is replaced with integration as follows (assuming the state space of X is $\Omega_X = \{x : x_{min} \leq x \leq x_{max}\}$):

$$\phi^{\downarrow \mathbf{X}'}(\mathbf{x}') = \phi^{-X}(\mathbf{x}') = \int_{\Omega_X} \phi(\mathbf{x})\, dx$$

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

Definition 4. (**Marginalization of Decision Variables.**) In this paper, all decision variables are discrete and binary. Assume I is a discrete decision variable with possible values $I = 0$ and $I = 1$ that has a continuous parent S with $\Omega_S = \{s : s_{min} \leq s \leq s_{max}\}$. Without loss of generality, arbitrarily assign $I = 0$ to the binary state of I that maximizes the value of u at s_{min}. To remove I from the ID, a threshold, Ψ, is determined as follows:

INPUT: $u, s_{min}, s_{max}, \epsilon$
OUTPUT: Ψ
INITIALIZATION: $\Psi = s_{min}$
DO WHILE $(u(I = 0, \Psi + \epsilon) \geq$
$u(I = 1, \Psi + \epsilon)) \cap (\Psi \leq s_{max})$
$\Psi = \Psi + \epsilon$
END DO
$\Psi = \Psi + \epsilon/2$

The parameter ϵ is an increment in S that can be assigned an appropriate value based on the application being addressed. The decision variable I is removed from the model by constructing the following MTE potential using the utility function u and the threshold value Ψ:

$$u^{\downarrow S}(s) = \begin{cases} u(I = 0, s) & \text{if } s_{min} \leq s < \Psi \\ u(I = 1, s) & \text{if } \Psi \leq s \leq s_{max} \end{cases}.$$

This definition applies when $u(I = 0, s) = u(I = 1, s)$ at one point and is a simpler version of the line search technique defined by Cobb and Shenoy (2008).

2.4 Fusion Algorithm

The ID is solved by applying the fusion algorithm (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 (according to Definition 2), then the variable is marginalized from the result. The appropriate marginalization operation depends on whether the variable being marginalized is a chance variable (see Definition 3) or a decision variable (see Definition 4).

3 ID Model

This section describes the ID model.

3.1 Graphical Representation

The ID model for the credit card detection problem is shown in Figure 1. The single-border ovals represent discrete chance variables. Fraud (F) indicates whether or not an order is fraudulent. The variables A_0 and P_0 reveal the number of suspicious characteristics in the address and product information on an order, respectively. The double-border oval for order Size (S) defines a continuous chance variable for the value (or cost) of the merchandise contained on an order. The arrows (or *arcs*) pointing from F to A_0, F to P_0, and F to S specify that the probability distributions for those variables are conditioned on F.

The rectangle in Figure 1 represents the firm's decision on whether or not to investigate an order. The arcs pointing from the variables A_0 to

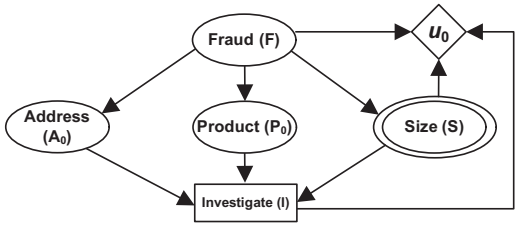


Figure 1: ID Model.

I, P_0 to I, and S to I show that the firm will observe the values of these chance variables prior to making its decision on whether or not to investigate the order. The value $I = 0$ means that the firms does not investigate, while the value $I = 1$ means the firm investigates.

The diamond in the ID represents the joint utility function. The arcs pointing from F, S, and I to this node indicate that these variables are in the domain of the joint utility function. In this context, the firm's utility will be the cost to the firm of either investigating potential fraud or shipping unpaid merchandise.

3.2 Numerical Representation

This section describes the potentials in the ID.

3.2.1 Fraud (F)

The variable F has state space $\Omega_F = \{0, 1\}$, where $F = 0$ stands for legitimate and $F = 1$ signifies fraudulent. Thus, the probability of fraud is denoted by $P(F = 1) = \eta$.

3.2.2 Address Characteristics (A_0)

The variable A_0 has state space $\Omega_{A_0} = \{0, \ldots, m\}$. The distribution $P(A_0|F)$ is constructed by using m variables representing the presence of specific address characteristics for credit card orders. For illustrative purposes, the remainder of the description of the potentials in the model will assume $m = 3$. Extension to the more general case is straightforward.

The variable A_0 "aggregates" the factors represented by the variables $\{A_1, A_2, A_3\}$; thus, the value of A_0 is determined by the number of variables in the set $\{A_1, A_2, A_3\}$ whose values equal one. The distribution $P(A_0|(A_1, A_2, A_3))$ is shown in Table 1. Combining the information in the variables $\{A_1, \ldots, A_m\}$ into one variable (A_0) reduces the number of decision rules determined when solving the ID from 2^{m+n} to $(m + 1)(n + 1)$, which significantly reduces the computational complexity of the solution. The resulting policies are also easier to implement.

Table 1: $P(A_0|(A_1, A_2, A_3))$.

			A_0			
A_1	A_2	A_3	0	1	2	3
0	0	0	1	0	0	0
1	0	0	0	1	0	0
0	1	0	0	1	0	0
0	0	1	0	1	0	0
1	1	0	0	0	1	0
0	1	1	0	0	1	0
1	0	1	0	0	1	0
1	1	1	0	0	0	1

The probability potential for the factor A_i given F is defined according to two parameters. The probabilities of the presence of the address inconsistency given the two states of F are $\alpha_{i,0} = P(A_i = 1|F = 0)$ and $\alpha_{i,1} = P(A_i = 1|F = 1)$, $\alpha_{i,0} < \alpha_{i,1}$, for $i = 1, 2, 3$.

Given $P(A_0|(A_1, A_2, A_3))$ and $P(A_i|F)$ for $i = 1, 2, 3$, the probabilities $P(A_0 = i|F = f)$ are determined as

$$P(A_0|F) = (P(A_0|(A_1, A_2, A_3)) \otimes P(A_1|F) \otimes P(A_2|F) \otimes P(A_3|F))^{-\{A_1,A_2,A_3\}}$$

according to Definitions 2 and 3. The result is shown in Table 2 and follows directly from the chain rule for Bayesian networks (Pearl, 1988).

3.2.3 Product Characteristics (P_0)

The variable P_0 has state space $\Omega_{P_0} = \{0, \ldots, n\}$. The distribution $P(P_0|F)$ is constructed by using n variables representing the presence of specific product characteristics for credit card orders. The variables $P_1, \ldots, P_n$ are factors related to an order's product information that may be useful for distinguishing a legitimate order from a fraudulent order. For example, fraudulent orders are more likely than acceptable orders to have multiples of the same item.

Table 2: Probability Distribution for A_0 given F ($P(A_0 = i|F = f)$).

	$F = 0$	$F = 1$
$A_0 = 0$	$(1 - \alpha_{10})(\alpha_{20} - 1)(\alpha_{30} - 1)$	$(1 - \alpha_{11})(\alpha_{21} - 1)(\alpha_{31} - 1)$
$A_0 = 1$	$\alpha_{20} + \alpha_{30} - 2\alpha_{20}\alpha_{30}$ $+\alpha_{10}(1 - 2\alpha_{30} + \alpha_{20}(3\alpha_{30} - 2))$	$\alpha_{21} + \alpha_{31} - 2\alpha_{21}\alpha_{31}$ $+\alpha_{11}(1 - 2\alpha_{31} + \alpha_{21}(3\alpha_{31} - 2))$
$A_0 = 2$	$\alpha_{20}\alpha_{30} + \alpha_{10}(\alpha_{20} + \alpha_{30} - 3\alpha_{20}\alpha_{30})$	$\alpha_{21}\alpha_{31} + \alpha_{11}(\alpha_{21} + \alpha_{31} - 3\alpha_{21}\alpha_{31})$
$A_0 = 3$	$\alpha_{10}\alpha_{20}\alpha_{30}$	$\alpha_{11}\alpha_{21}\alpha_{31}$

The probabilities of the presence of the product characteristics given the two states of F are $\rho_{j,0} = P(P_j = 1|F = 0)$ and $\rho_{j,1} = P(P_j = 1|F = 1)$, $\rho_{j,0} < \rho_{j,1}$, $j = 1, \ldots, n$. The variable P_0 summarizes the information in $\{P_1, \ldots, P_n\}$ in much the same way as A_0 summarizes the information in the address characteristics for an order.

Calculation of $P(P_0|F)$ is accomplished in the same way as the determination of $P(A_0|F)$, so the details are omitted. More information is provided in (Cobb, 2010).

3.2.4 Order Size (S)

The chance variable S has $\Omega_S = \{s : s_{min} \leq s \leq s_{max}\}$. The probability potential ϕ for $\{F, S\}$ represents the conditional probability density functions for S given $F = 0$ and $F = 1$.

The parameters used in the example of the next section will be used to describe the potential ϕ. Suppose the natural log of S is normally distributed with mean $\mu = 2.5$ and variance $\sigma^2 = 0.5$ given that $F = 0$, i.e. $S|F = 0 \sim LN(2.5, 0.5)$. Also, assume the natural log of S given $F = 1$ is normally distributed with $\mu = 3.5$ and $\sigma^2 = 0.75$, i.e. $S|F = 1 \sim LN(3.5, 0.75)$. The MTE potential fragment representing the conditional distribution for S given $F = 0$ is defined as

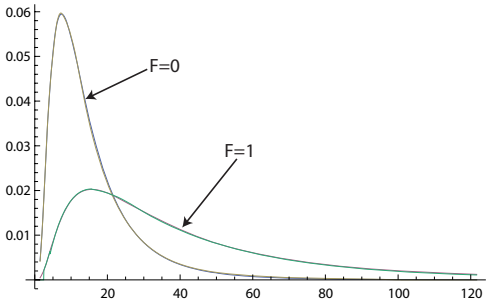


Figure 2: MTE Lognormal Approximations.

$$\phi(F = 0, s) = \hat{f}_{F=0}(s) =$$

$$\begin{cases} 39.78 + 5.71\ \exp\{0.0326(s - 7.39)\} \\ -45.34\ \exp\{0.0032(s - 7.39)\} \\ \qquad \text{if } 1.46 \leq s < 2.72 \\ 21.72 - 1.49\ \exp\{-0.0468(s - 7.39)\} \\ -20.17\ \exp\{0.0035(s - 7.39)\} \\ \qquad \text{if } 2.72 \leq s < 7.39 \\ \vdots \end{cases}$$

The full description of the function can be found in (Cobb, 2010). The MTE potential fragment $\phi(F = 1, s)$ representing the conditional distribution for S given $F = 1$ is defined similarly, and both MTE potentials are displayed in Figure 2, overlaid on the actual lognormal distributions. The prior distribution for order size is skewed farther to the right for fraudulent orders.

3.2.5 Utility Function (u_0)

The joint utility function u_0 has domain $\{F, S, I\}$. Assume that the cost of investigating an order for fraudulent activity is c (a constant). The values for u_0 are $u_0(F = 0, I = 0, s) = 0$,

$u_0(F = 0, I = 1, s) = -c$, $u_0(F = 1, I = 0, s) = -s$, and $u_0(F = 1, I = 1, s) = -c$.

If the firm investigates, it incurs the cost of the investigation, but avoids forfeiting the value of the merchandise when a fraudulent order is thwarted. If the firm fails to investigate a fraudulent order, it incurs a cost equal to the value of the merchandise shipped to fulfill the order.

The joint utility function u_0 is approximated by the MTE potential u_1, which is identical to u_0 with the exception of one term, which is defined (Cobb and Shenoy, 2006) as $u_1(F = 1, I = 0, s) = (s_{max} - s_{min})(13.512870 \cdot \exp\left\{\frac{0.071387}{s_{max}-s_{min}} \cdot (s - s_{min})\right\} - 13.507018) - s_{min}$.

For the case where $F = 1$ and $I = 0$, the function u_1 is an MTE approximation to the linear function $g(s) = -s$. In the ID solution process, the MTE utility function will be combined via multiplication with the MTE density potential for S given F. Since the class of MTE potentials is closed under addition and multiplication, the result remains an MTE potential. This allows the resulting function to be integrated in closed form to determine the firm's maximum expected utility.

4 Example

This section describes an example where optimal decision rules are developed that allow the firm to decide when to investigate potentially fraudulent orders.

4.1 Problem Description

Assume $m = n = 3$, meaning that there are three address factors and three product factors used to determine the conditional distributions for A_0 given F and P_0 given F, respectively. These factors are:

Shipping and billing addresses match (A_1)

Untraceable e-mail (A_2) — the order originated from a free, web-based address.

Foreign address (A_3)

Leave at home (P_1) — the customer requests that the shipment be left at the door if no one is home.

Rush shipping (P_2)

Multiple units of the same item (P_3)

The presence of these factors is denoted by either $A_i = 1$ or $P_j = 1$ and corresponds to a higher incidence of fraud.

The potential representing the prior probability distribution for F has values $P(F = 0) = 1 - \eta = 0.99$ and $P(F = 1) = \eta = 0.01$. The conditional probability density functions for order Size (S) are those approximated by the MTE potential ϕ in Figure 2. The cost of investigating an order is $c = 10$, and the MTE potential fragment approximating $u_0(F = 1, I = 0, s)$ in the joint utility function u_0 is $u_1(F = 1, I = 0, s) = 5989.45 - 5993.51 \exp\{0.000161(s - 1.46)\}$.

The probability distribution $P(A_0|F)$ is determined using the result in Table 2, and $P(P_0|F)$ is calculated similarly. A summary of the parameters in the example problem is given in Table 3.

Table 3: Parameters for the Example.

Variable	$F = 0$	$F = 1$
Fraud (F)	$1 - \eta = .99$	$\eta = .01$
Match (A_1)	$\alpha_{10} = .25$	$\alpha_{11} = .40$
Email (A_2)	$\alpha_{20} = .01$	$\alpha_{21} = .05$
Inter. (A_3)	$\alpha_{30} = .05$	$\alpha_{31} = .25$
Leave (P_1)	$\rho_{10} = .20$	$\rho_{11} = .30$
Rush (P_2)	$\rho_{20} = .10$	$\rho_{21} = .20$
Mult. (P_3)	$\rho_{30} = .05$	$\rho_{31} = .075$
Size (S)	$LN(2.5, .5)$	$LN(3.5, .75)$

4.2 Solution

This section briefly describes the solution to the example using the fusion algorithm. In this problem, a possible deletion sequence is F, I, S, A_0, P_0.

The potentials in the model at the outset are $P(F)$, $P(A_0|F)$, $P(P_0|F)$, ϕ for $\{S, F\}$, and u_1 for $\{F, S, I\}$. The first variable in the deletion sequence is F, and since all potentials contain F in their domain, all must be combined prior to the marginalization of F. The combination

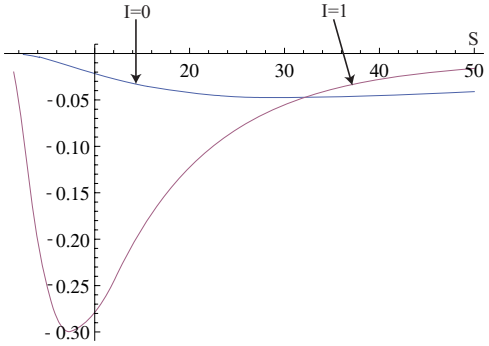


Figure 3: u_2 where $A_0 = 2$ and $P_0 = 2$.

results in an MTE utility potential determined as

$$u_2' = P(F) \otimes P(A_0|F) \otimes P(P_0|F) \otimes \phi \otimes u_1 \ .$$

The variable F is marginalized as

$$\begin{aligned} u_2(A_0 = i, P_0 = j, I = k, s) = \\ u_2'(F = 0, A_0 = i, P_0 = j, I = k, s) \\ + u_2'(F = 1, A_0 = i, P_0 = j, I = k, s) \end{aligned}$$

for all $(i, j, k, s) \in \Omega_{\{A_0, P_0, I, S\}}$. The function u_2 is shown graphically in Figure 3 for the case where the number of observed address inconsistencies and suspicious product characteristics are both two ($A_0 = 2$ and $P_0 = 2$). The expected utility that results from investigating the potential fraud ($I = 1$) is less than the expected utility associated with not investigating ($I = 0$) for smaller orders.

The objective of the firm is to decide optimally whether to investigate potential fraud after it observes the values of A_0, P_0, and S. Thus, for each configuration of states of the discrete variables $A_0 = i$ and $P_0 = j$, the firm must choose a threshold $\Psi_{i,j}$ for order size using the procedure in Definition 4. As an example, $\Psi_{2,2} = 31.96$ and is determined by finding the point where the two functions in Figure 3 are approximately equal. For values of S below this threshold, the firm will be better off in the long run not investigating the order for potential fraud. For values of S above this threshold, due to the potential loss of merchandise, the firm should investigate potential fraud on an order. Methods of investigating fraud include validating the billing address, shipping address, e-mail address, and phone number, and contacting the customer to confirm the order. This investigation is carried out at an average cost of $c = 10$ per order.

Table 4: Decision Thresholds $\Psi_{i,j}$.

Ψ	$P_0 = 0$	$P_0 = 1$	$P_0 = 2$	$P_0 = 3$
$A_0 = 0$	95.56	89.86	76.96	61.96
$A_0 = 1$	83.46	67.16	52.56	44.06
$A_0 = 2$	47.16	38.96	31.96	26.86
$A_0 = 3$	28.16	23.16	19.06	16.16

The decision variable I is removed from the model by constructing the following MTE potential using the utility function u_2 and the threshold values $\Psi_{i,j}$:

$$\begin{aligned} u_3(A_0 = i, P_0 = j, s) = \\ \begin{cases} u_2(A_0 = i, P_0 = j, I = 0, s) \\ \qquad \text{if } s_{min} \leq s < \Psi_{i,j} \\ u_2(A_0 = i, P_0 = j, I = 1, s) \\ \qquad \text{if } \Psi_{i,j} \leq s \leq s_{max} \end{cases} . \end{aligned}$$

for $i = 0, 1, 2, 3$ and $j = 0, 1, 2, 3$.

4.3 Results

The decision thresholds $\Psi_{i,j}$ for the example problem are shown in Table 4. At higher numbers of address inconsistencies and suspicious product characteristics, the order size thresholds are lower, since the firm has more reason to believe that such orders are not legal.

The maximum expected utility from following the decision thresholds is calculated as

$$u_4 = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} \left(\int_{\Omega_S} u_3(A_0 = i, P_o = j, s) \, ds \right) \right) .$$

For the example problem, the maximum expected utility is $u_4 = -0.31173$, which represents the fraud investigation and loss expense stated on a per order basis.

Additional results and sensitivity analysis for this example are provided in (Cobb, 2010).

5 Conclusions

An ID model that can be used to detect potentially fraudulent online credit card transactions

was introduced. A business can employ such a model to establish decision policies that guide employees to investigate orders that are most likely to be suspect. By following such policies, a business can minimize its total fraud-related expenses, which include both the costs of lost merchandise and the expense of following up on suspicious orders.

In addition to considering order size as a criteria for identifying potentially fraudulent orders, the ID model allows a business to consider other characteristics of the address and product information on an order. When more of these factors indicate that fraud may be present, the order size threshold used to decide whether or not to investigate an order is lowered, because an illegal order becomes more likely. Using the ID to establish such rules allows a business to investigate the orders that are most likely to be fraudulent and save the cost of such inquiries on orders—even large ones—that are most likely legitimate.

Future research can incorporate additional complexities to make the model more realistic. For instance, an implicit assumption is that the investigation always concludes with certainty that an order is fraudulent. A node representing the result of the investigation can be added to the ID to relax this assumption. In cases where a good order is mistakenly canceled, the cost of the customer's dissatisfaction should be considered in the utility function. Also, the investigation cost, c, may not be a constant and can be modeled as a random variable, perhaps conditional on the number of suspicious address and product characteristics observed.

Acknowledgments

The author is grateful for the insightful comments of three anonymous reviewers. Support from the Spanish Ministry of Science and Innovation through project TIN2007-67418-C03-02 and by EFDR funds is gratefully acknowledged.

References

Cobb, B.R. 2010. Detecting Online Credit Card Fraud with Hybrid Influence Diagrams, Working Paper, Virginia Military Institute, Lexington, VA. Available for download at: `www.vmi.edu/fswebs.aspx?tid=24697&id=24791`

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. *European J. Oper. Res.* **186**(1) 261–275.

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

Credit card fraud: A guide to help businesses recognize it, report it, stop it. 2008. www.visa.ca/en/merchant/pdfs/merchant_fraud.pdf. Accessed on 26 November 2008.

Howard, R.A., J.E. Matheson. 1984/2005. 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.

Pearl, J. 1988. *Probabilistic Reasoning in Expert Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Francisco, CA.

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.

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.

Continuous Decision Variables with Multiple Continuous Parents

Barry R. Cobb
Virginia Military Institute
cobbbr@vmi.edu

Abstract

This paper introduces an influence diagram (ID) model that permits continuous decision variables with multiple continuous parents. The marginalization operation for a continuous decision variable first develops a piecewise linear decision rule as a continuous function of the next continuous parent in the deletion sequence. Least squares regression is used to convert this rule to a piecewise linear function of all the decision variable's continuous parents. This procedure is incorporated into an iterative solution algorithm that allows more refined decision rules to be constructed once the non-optimal regions of the state spaces of decision variables are identified. Additional examples serve to compare relative advantages of this technique to other ID models proposed in the literature.

1 Introduction

The *influence diagram* (ID) is a graphical and numerical representation for a decision problem under uncertainty (Howard and Matheson, 1984). 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. An example of a decision problem under uncertainty is given in the following section.

1.1 Example

A firm facing uncertain demand must choose production capacity and set product prices (Gök, 2002). Product demand is determined as $Q(p,z) = 12 - p + z$, where P is the product price and Z is a random demand "shock." Assume $Z \sim N(0,1)$ and that the firm's utility (profit) function is

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

Notice that the firm's sales are limited to the minimum of product demand and production capacity (K). Figure 1 shows an ID model for the example. The chance and decision variables in the ID are depicted as ovals and rectangles, respecitively. The joint utility function appears as a diamond. Since there is an arrow pointing from Z to K and P, Z is a *parent* of K and P. The set of all parents of P is $Pa(P) = \{K, Z\}$. An arrow pointing to a chance node indicates the distribution for this node is conditioned on the variable at the head of the arrow. An arrow pointing to a decision node means that the value of the variable will be known when the decision is made.

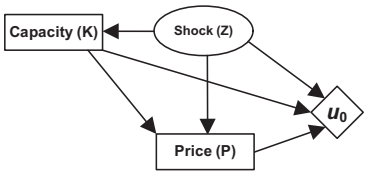


Figure 1: Influence Diagram Model.

1.2 Background

Although most ID models proposed in the literature assume that all decision variables take values in discrete (countable) state spaces, there are some exceptions. Shachter and Kenley (1989) introduce Gaussian IDs, where all continuous chance variables are normally dis-

Pp. 97–105 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

tributed, all decision variables are continuous, and utility functions are quadratic.

The mixture-of-Gaussians ID (Poland and Shachter, 1993) requires continuous chance variables to be modeled as mixtures of normal distributions and allows continuous decision variables. Madsen and Jensen (2005) outline an improved solution procedure for IDs constrained under the same conditions as mixture-of-Gaussians IDs that is able to take advantage of an additive factorization of the joint utility function.

Cobb (2007) introduces an ID model which allows continuous decision variables with one continuous parent and continuous chance variables having any probability density function (pdf). Using this approach, pdfs and utility functions are approximated by mixtures of truncated exponentials (MTE) potentials (Moral et al., 2001), which allows the marginalization operation for continuous chance variables to be performed in closed form. This technique develops a piecewise linear decision rule for continuous decision variables and subsequently marginalizes them from the model as deterministic chance variables.

This paper builds upon the model in (Cobb, 2007) by allowing continuous decision variables to have multiple continuous parents. The marginalization operation for continuous decision variables and the iterative solution algorithm are introduced using examples. A longer working paper (Cobb, 2010) contains more formal definitions.

The remainder of this paper is organized as follows. In §2, notation and definitions are introduced. In §3, a procedure for marginalizing a continuous decision variable is presented using the example in §1.1. In §4, the results from the example problem are compared to an analytical solution. §5 describes solutions to additional examples before §6 concludes the paper.

2 Notation and Definitions

2.1 Notation

In this paper, we assume all decision and chance variables take values in finite-bounded, continuous (non-countable) state spaces. All variables are denoted by capital letters in plain text, e.g., A, B, C. Sets of variables are denoted by capital letters in boldface, with $\mathbf{Z}$ representing chance variables, $\mathbf{D}$ representing decision variables, and $\mathbf{X}$ indicating a set of variables whose components are a combination of chance and decision variables. If A and $\mathbf{X}$ are one- and multi-dimensional variables, respectively, then a and $\mathbf{x}$ represent specific values of those variables. The finite-bounded, continuous state space of $\mathbf{X}$ is denoted by $\Omega_{\mathbf{X}}$.

Example 1. *In the ID shown in Figure 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* $\Omega_Z = \{z : -3 \leq z \leq 3\}$. *This assumes the distribution for* Z *is normalized over the interval* $[-3, 3]$ *to solve the ID.*

MTE probability potentials are denoted by lower-case Greek letters, e.g., ϕ, ψ, φ, whereas MTE utility potentials are denoted by u_i, where the subscript i is normally zero for the joint utility function in the problem, and one for the initial MTE approximation to the joint utility function. The subscript can be increased to index additional MTE utility potentials in the initial representation or solution.

2.2 Mixtures of Truncated Exponentials (MTE) Potentials (Moral et al., 2001)

Let $\mathbf{X}$ be a mixed variable and let $\mathbf{Z} = (Z_1, \ldots, Z_c)$ and $\mathbf{D} = (D_1, \ldots, D_f)$ be the chance and decision variable parts of $\mathbf{X}$, respectively. Given a partition $\Omega_1, \ldots, \Omega_n$ that divides $\Omega_{\mathbf{X}}$ into hypercubes, an n-piece MTE potential $\phi : \Omega_{\mathbf{X}} \mapsto \mathcal{R}^+$ has components

$$\phi_h(\mathbf{z}, \mathbf{d}) = a_0 + \sum_{i=1}^{m} a_i \exp\left\{ \sum_{j=1}^{c} b_i^{(j)} z_j + \sum_{\ell=1}^{f} b_i^{(c+\ell)} d_\ell \right\}$$

for $h = 1, \ldots, n$, where $a_i, i = 0, \ldots, m$ and $b_i^{(j)}$, $i = 1, \ldots, m$, $j = 1, \ldots, (c+f)$ are real numbers.

We assume all MTE potentials are equal to zero in unspecified regions. In this paper, all probability distributions and utility functions are approximated by MTE potentials.

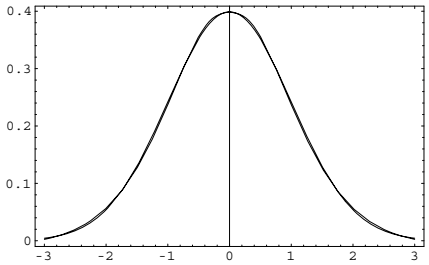


Figure 2: $N(0,1)$ pdf and MTE Potential ϕ_1.

Example 2. *The function $f_1(p) = p$ over the interval $[1, 9]$ can be approximated by the MTE potential*

$$u_P(p) = \begin{cases} -53.028072 + 54.05148 \exp\{0.017847(p-1)\} \\ \qquad \text{if } 1 \leq p < 5 \\ -49.028072 + 54.05148 \exp\{0.017847(p-5)\} \\ \qquad \text{if } 5 \leq p \leq 9\,, \end{cases}$$

using the method described in (Cobb and Shenoy, 2008). Similar MTE potentials, $u_K(k)$ and $u_Z(z)$, are used to approximate the functions $f_2(k) = k$ on $[0, 14]$ and $f_3(z) = z$ on $[-3, 3]$. These approximations are are substituted into (1) to form the MTE utility function u_1 for the example problem.

The resulting function will contain values of variables in the limits of the domain. In other words, to create a true MTE potential where the limits are hypercubes, values for two of the variables must be substituted. MTE potentials defined in this way require replacement of linear terms when integration is used to marginalize variables in the ID solution. This is discussed in (Cobb and Shenoy, 2006).

Example 3. *The MTE approximation ϕ_1 to the $N(0, 1)$ pdf (see Cobb et al., (2006) for numerical details) that approximates the distribution for the variable Z in the example from §1.1 is shown in Figure 2, overlaid on the actual $N(0, 1)$ distribution. The MTE function is normalized on the interval $[-3, 3]$.*

2.3 Fusion Algorithm

IDs are solved in this paper by applying the fusion algorithm of Shenoy (1993), which is relevant for the case where the joint utility function factors multiplicatively. 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 via pointwise multiplication, then the variable is marginalized from the result. The appropriate marginalization operation depends on whether the variable being marginalized is a chance variable (in which case marginalization is accomplished by integrating over the domain of the chance variable being removed) or a decision variable. Formal definitions of combination and marginalization of chance variables can be found in (Cobb, 2010).

3 Marginalizing Decision Variables

Assume we want to eliminate a decision variable D with parents $\mathbf{X} = \{X_1, X_2 \ldots, X_n\}$ from the ID. The variables in $\mathbf{X}$ may be either chance or decision variables, and the subscripts on variables in $\mathbf{X}$ serve to number the variables as they appear in the deletion sequence for the problem. The set of parents excluding X_1 is denoted by $\mathbf{X}' = \mathbf{X} \setminus X_1$. Eliminating the decision variable is a four-step process:

(1) Combine all potentials containing D in their domain, create discrete approximations to Ω_D and $\Omega_{\mathbf{X}'}$, and find the discrete value of D that maximizes utility for each region of a hypercube of Ω_{X_1} for each (discrete) $\mathbf{x}' \in \Omega_{\mathbf{X}'}$.

(2) For each (discrete) $\mathbf{x}' \in \Omega_{\mathbf{X}'}$, create a decision rule for D as a piecewise linear function of X_1.

(3) Use least squares regression to create a piecewise linear decision rule for D as a function of $\mathbf{X}$.

(4) Convert D to a deterministic chance variable, and marginalize D using the procedure in §3.4.

This process has similarities to the procedure for marginalizing a continuous decision variable proposed by Cobb (2007); however, employing regression in Step 3 enables this new operation to permit continuous decision variables with multiple continuous parents. The steps are introduced by illustrating the removal of P from the ID of §1.1 using the deletion sequence P, K, Z.

In this solution, we utilize $v = 8$ discrete values and regions at each step in the process when we are required to discretize or sub-divide the state space of a continuous variable.

3.1 Step 1–Discrete Approximation

The purpose of this step in the marginalization process is to find a relationship—given a value of Z—between the optimal price (P) and production capacity (K) by examining the utility function for various values of P.

In this step, discrete values p_u, $u = 1, \ldots, 8$, for P are assigned as $\{1.5, 2.5, \ldots, 8.5\}$. Assign discrete values z_t, $t = 1, \ldots, 8$, to the chance variable Z, the most distant parent of P in the deletion sequence. Based on the state space Ω_Z, these discrete values are $\{-2.625, -1.875, -1.125, \ldots, 2.625\}$. For each discrete value z_t, create an MTE utility function $u_1(k, p, z_t)$ by substituting $Z = z_t$ in u_1.

For each value z_t, determine the (discrete) value in Ω_P that maximizes the utility function $u_1(k, p, z_t)$ for each region of a hypercube of Ω_K. For example, when $Z = z_3 = -1.125$, the utility functions $u_1(k, p_u, z_3)$ appear as shown in Figure 3. From the diagram, it is apparent that $u_1(k, 8.5, z_3) \approx u_1(k, 7.5, z_3)$ when $K = 2.75$, $u_1(k, 7.5, z_3) \approx u_1(k, 6.5, z_3)$ when $K = 4.05$, and $u_1(k, 6.5, z_3) \approx u_1(k, 5.5, z_3)$ when $K = 5.35$.

The results of this step of the operation are the sets of points, $\Phi_{1,t}$, and decision variable values, $\Psi_{1,t}$, for $t = 1, \ldots, 8$. For instance, $\Phi_{1,3} = \{0, 2.75, 4.05, 5.35, 14\}$ and $\Psi_{1,3} = \{8.5, 7.5, 6.5, 5.5\}$, where the three in the subscripts is an index on the related value $z_3 = -1.125$. The set $\Phi_{1,3}$ can be used to determine intervals where the optimal discrete value of price is invariant, and the set $\Psi_{1,3}$ contains the optimal values for P corresponding to these intervals.

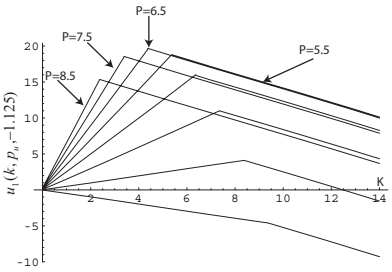


Figure 3: The Utility Functions $u_1(k, p_u, z_3)$.

This procedure is derived from the operation for marginalizing a discrete decision variable in a hybrid ID (Cobb and Shenoy, 2008).

3.2 Step 2–Piecewise Linear Decision Rule

The purpose of this step is to express the relationship between optimal price (P) and production capacity (K) by estimating a continuous function $P = f(K)$, given a value for Z.

Continuing from §3.1, when $-1.5 \leq z \leq -0.75$, $\Phi_{1,3}$ is used to determine $k = \left(\frac{0+2.75}{2}, \frac{2.75+4.05}{2}, \frac{4.05+5.35}{2}, \frac{5.35+14}{2}\right) = (1.375, 3.4, 4.7, 9.675)$, with a corresponding set of points, $p = (8.5, 7.5, 6.5, 5.5)$ defined as in $\Psi_{1,3}$. The equation for the line connecting the coordinates $\{(k = 1.375, p = 8.5), (k = 3.4, p = 7.5)\}$ is $p(k) = 9.17901 - 0.49383k$. Similar equations are determined using other sets of adjacent coordinates and these form a piecewise linear decision rule for P as

$$\hat{\Psi}_{1,3}(k) = \begin{cases} 9 & 0 \leq k < 0.3625 \\ 9.17901 - 0.49383k & 0.3625 \leq k < 3.4 \\ 10.11540 - 0.76923k & 3.4 \leq k < 4.7 \\ 7.44472 - 0.20101k & 4.7 \leq k \leq 14\,. \end{cases}$$

The equation for the first (last) line segment is extrapolated until the result of the function is greater (less) than the endpoint of Ω_P, in which

case the function is defined as the maximum (minimum) value in Ω_P. A similar decision rule is developed for the regions with midpoints z_t, $t = 1, \ldots, 8$. A function $\hat{\Psi}_1$ is comprised of the resulting piecewise functions as

$$\hat{\Psi}_1(k,z) = \begin{cases} \hat{\Psi}_{1,1}(k) & -3 \le z < -2.25 \\ \vdots & \vdots \\ \hat{\Psi}_{1,8}(k) & 2.25 \le z \le 3\ . \end{cases}$$

3.3 Step 3–Least Squares Regression

This step further refines the decision rule to be a compact piecewise linear function for optimal price given values of K and Z.

Continuing from §3.2, the state space of K—the next parent of P in the deletion sequence—is divided into 8 regions, $[0, 1.75], \ldots, [12.25, 14]$, with the m-th region denoted by $[k^d_{m-1}, k^d_m]$ where $k^d_m = k_{min} + m \cdot (k_{max} - k_{min})/v$ for $m = 0, \ldots, v$. Define $\varepsilon = 0.1$, $n_K = \lfloor (k_{max} - k_{min})/\varepsilon \rfloor + 1$, and $n_Z = \lfloor (z_{max} - z_{min})/\varepsilon \rfloor + 1$. The function $\hat{\Psi}_1$ is used to output a series of ordered data points $\{\hat{\Psi}_1(k_i, z_j), k_i, z_j\}$ for each $k_i = k_{min} + (i-1) \cdot \varepsilon$, $i = 1, \ldots, n_K$ and $z_j = z_{min} + (j-1) \cdot \varepsilon$, $j = 1, \ldots, n_Z$. These ordered data points are sorted into ascending order according to the values k_i and grouped into $v = 8$ tables, where the m-th table contains points such that all $k_i \in [k^d_{m-1}, k^d_m]$ for each $m = 1, \ldots, v$. In other words, for each value k_i that appears in the m-th table, each pair (k_i, z_j) appears exactly once, along with the corresponding values $\hat{\Psi}_1(k_i, z_j)$. Each table is used to create the matrices required to estimate a linear equation $\hat{p}(k,z) = b_{2m} + b_{3m} \cdot k + b_{4m} \cdot z$ via least squares regression.

For example, with $\varepsilon = 0.1$, $n_K = 141$, and $n_Z = 61$, so 61 values for Z are matched with each of the 18 values of K in the second interval, $[1.75, 3.5]$, defined using Ω_K. Thus, $18 \times 61 = 1098$ data points are used to define the (1098×1) matrix Υ_2 and the (1098×3) matrix Λ_2 as follows:

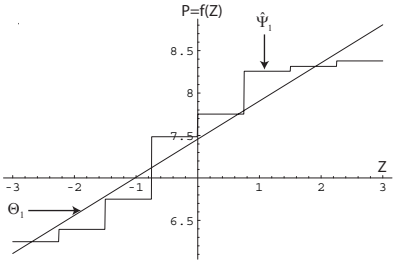


Figure 4: The Decision Rule $\Theta_1(4.375, z)$ for P.

$$\Upsilon_2 = \begin{bmatrix} \hat{\Psi}_1(k_{19}, z_1) \\ \vdots \\ \hat{\Psi}_1(k_{19}, z_{61}) \\ \vdots \\ \hat{\Psi}_1(k_{36}, z_1) \\ \vdots \\ \hat{\Psi}_1(k_{36}, z_{61}) \end{bmatrix} \quad \Lambda_2 = \begin{bmatrix} 1 & k_{19} & z_1 \\ \vdots & \vdots & \vdots \\ 1 & k_{19} & z_{61} \\ \vdots & \vdots & \vdots \\ 1 & k_{36} & z_1 \\ \vdots & \vdots & \vdots \\ 1 & k_{36} & z_{61} \end{bmatrix}$$

The least squares regression estimators are determined as $\mathbf{b}_2 = [\ b_{22}\ \ b_{32}\ \ b_{42}\ ]^\top = \left(\Lambda_2^\top \Lambda_2\right)^{-1} \Lambda_2^\top \Upsilon_2$. Following this process in each region of the state space of K creates the following piecewise linear decision rule:

$$\Theta_1'(k,z) = \begin{cases} 9.0194 - 0.3631k + 0.0986z & 0 \le k < 1.75 \\ 9.0265 - 0.3808k + 0.3246z & 1.75 \le k < 3.5 \\ \vdots & \vdots \\ 8.1670 - 0.1731k + 0.5340z & 12.25 \le k \le 14\ . \end{cases}$$

A revised piecewise linear decision rule is then determined as

$$\Theta_1(k,z) = \begin{cases} p_{min} & \Theta_1'(k,z) < p_{min} \\ \Theta_1'(k,z) & p_{min} \le \Theta_1'(k,z) \le p_{max} \\ p_{max} & \Theta_1'(k,z) > p_{max}\ . \end{cases}$$

Using this revised formulation of the decision rule ensures that the assigned values are contained in Ω_P. A graphical view of the decision rule for P as a function of Z given that $K = 4.375$ is shown in Figure 4. The decision rule Θ_1 is a refinement of the decision rule $\hat{\Psi}_1$.

3.4 Step 4–Removing the Decision Variable

Continuing from §3.3, since a value for P will be completely determined by observed values of K and Z, P can be replaced in the joint utility function as $u_2(k,z) = u_1(\Theta_1(k,z),k,z)$. The substitution of Θ_1 for P in u_1 is accomplished on a piecewise basis. For instance, when $5.25 \leq k \leq 7$, Θ_1 is defined as $f_1(k,z) = 8.16701 - 0.17307k + 0.47450z$ for all $z \in \Omega_Z$. When $k \geq 12 - p + z$, $0 \leq z \leq 3$, and $1 \leq p \leq 5$, u_1 is defined as

$$f_2(p,k,z) = -1067.4336 - 94.5901\exp\{0.0102k\} + \cdots + 2311.6851\exp\{0.0179p + 0.02380z\} + \cdots$$

The calculation of $u_2(k,z) = u_1(\Theta_1(k,z),k,z)$ includes the result of the substitution $f_2(f_1(k,z),k,z)$, with the ensuant expression included in u_2 where the domains of the two functions overlap, or

$$f_2(f_1(k,z),k,z) = -1067.4336 - 94.5901\exp\{0.0102k\} + \cdots + 2311.6851\exp\{0.0179 \cdot f_1(k,z) + 0.02380z\} + \cdots$$

for $5.25 \leq k \leq 7$ and $0.8269k - 0.5255z \geq 3.8330$. A similar substitution of each piece of Θ_1 is made into each piece of u_1 to create the MTE utility function u_2.

3.5 Results

To complete the example problem, the decision variable K is marginalized using the process in §3.1 through 3.4, except that since K has only one parent (Z), Step 3 (least squares regression) is not performed. The decision rule for K as a function of Z is determined as $\Theta_2(z) = 5.2646 + 0.5833z$ for all $z \in \Omega_Z$. To marginalize K, a new utility function u_3 is determined as $u_3(z) = u_2(\Theta_2(z),z)$. The firm's expected utility is then calculated as $\int_{\Omega_Z} \phi_1(z) \cdot u_3(z)\, dz = 24.6394$.

The ID method presented in this paper is sensitive to the state spaces assigned to the decision variables. In other words, if the continuous interval of possible optimal values for the decision variables can be narrowed, the accuracy of the decision rules can be improved.

In this example, the ID decision rule $\Theta_2(z)$ only selects values for K in the interval $[3.5146, 7.0146]$. Similarly, the decision rule $\Theta_1(k,z)$ only allows for values of P in the interval $[4.142, 9]$. In a second iteration of the ID solution procedure, we can replace the original state spaces of the decision variables P and K with these intervals and obtain a better approximation to the true optimal decision rules and profit function.

To complete the second iteration for the example, the same marginalization procedure is used to develop decision rules for P as a function of $\{K, Z\}$ and K as a function of Z.

Cobb (2009) explains additional details of the iterative algorithm.

4 Comparison

In the example problem, the firm knows the true value, $Z = z$, at the time it chooses capacity. Gök (2002) uses this fact to find an analytical solution for the optimal capacity of $k^*(z) = \frac{10+z}{2}$. This result hinges on several restrictive assumptions, including the linearity of the demand function and the symmetric form of the distribution for Z. By choosing an example with an analytical solution, we can compare the results from the ID solution as a means of determining its accuracy. The ID method can then be extended to cases where an analytical solution is not available (see §5).

The decision rule Θ_2 for K determined using two iterations of the ID solution procedure is shown in Figure 5 with the analytical capacity decision rule. This decision rule has seven linear pieces. The mean squared error (MSE) (Winkler and Hays, 1970) can be used as a measure of the difference between the analytical and estimated decision rules. The MSE is calculated as

$$\int_{\Omega_Z} \phi_1(z) \cdot (\Theta_2(z) - k^*(z))^2 \, dz = 0.01469\,.$$

The MSE after the first iteration is 0.07682, so revising the state space and performing the second iteration improves the accuracy of the decision rule. The decision rule Θ_1 for P is used in the determination of Θ_2, so this MSE measurement is a measure of the accuracy of the decision rules developed the ID solution. The

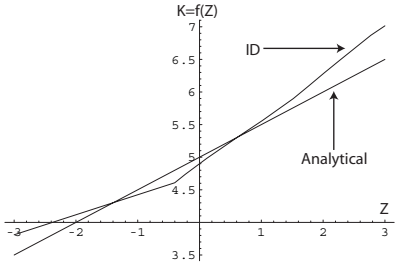


Figure 5: Decision Rules for $K = f(Z)$.

expected profit is 25.0792, as compared to the first iteration and analytical values of 24.6394 and 25.25, respectively.

5 Additional Examples

This section briefly describes two additional examples derived from the problem in §1.1 (for additional details, see (Cobb, 2010)).

5.1 Non-Gaussian Chance Variable

One advantage of using the ID model described in this paper is that it can accommodate non-Gaussian chance variables directly without using a mixture-of-Gaussians representation.

Suppose that the firm has established production capacity at a minimum of 3.5 units and a maximum of 6.5 units. The random variable K represents the percentage of additional capacity (above minimum) available (which fluctuates with changes in labor and machine utilization) and is modeled with a $Beta(3, 3)$ distribution. The distribution for K is approximated by the MTE potential ϕ_2 determined using the method discussed by Cobb et al. (2006). The MTE approximation ϕ_1 to the distribution for Z remains the same.

Although P now has two parents (K and Z) that are chance variables (one of which is non-Gaussian), the procedure for marginalizing P from the ID proceeds in exactly the same way as in the previous example.

5.2 Nonmonotonic Decision Rule

Suppose P and K are decision variables as in §1.1, but that the unit variable cost of \$1 is replaced in the joint utility function by z^2, i.e. unit variable costs are now higher for values of the demand shock Z farther from zero. This utility function is approximated with an MTE utility function as in Example 2.

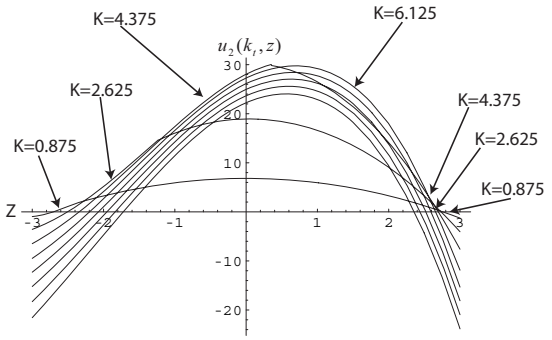


Figure 6: The Utility Functions $u_2(k_t, z)$ in the Example with Revised Unit Variable Cost.

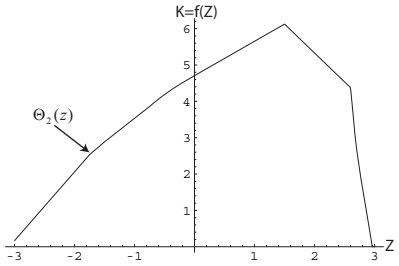


Figure 7: The Decision Rule Θ_2 for K.

Figure 6 shows the utility function (for eight discrete values of K) after marginalizaton of P and illustrates that the optimal value for K must be determined as a nonmonotonic function of Z. For instance, when $-2.25 \leq z \leq -1.15$ or $2.65 \leq z \leq 2.75$, a capacity of $K = 2.625$ is optimal, whereas if $-1.15 \leq z \leq 0.45$ or $2.55 \leq z \leq 2.65$, the best value of K is 4.375. Ultimately, these points are used to create the decision rule Θ_2 for K as a function of Z (see Figure 7).

6 Conclusions

This paper has introduced improvements to the model of Cobb (2007) that allow continuous decision variables in IDs to have multiple continuous parents. The framework proposed in this paper has some potential advantages over other ID models. To use the model in (Cobb, 2007) to solve the example in §1.1, we would have to impose one of the following restrictions: (1) model

K as a discrete decision variable or Z as a discrete chance variable, as the continuous decision variable P would be allowed to have only one continuous parent; or (2) discretize any pair of chance and/or decision variables. A comparison of the model in this paper to related models is provided in (Cobb, 2010).

The method in this paper permits non-Gaussian pdfs to be modeled without using mixtures of Gaussian distributions. This is in constrast to Gaussian IDs (Shachter and Kenley 1989) and mixtures-of-Gaussians IDs (Poland and Shachter, 1993; Madsen and Jensen, 2005). Additionally, those models determine only linear—as opposed to piecewise linear—decision rules, and thus cannot accomodate a case where the optimal decision rule is a nonmonotonic function of a decision rule's continuous parent(s), as in the example of §5.2.

Additional research is needed to demonstrate potential applications of the ID model and explain the compromise between computational cost and decision rule accuracy when parameters in the solution technique are altered. The model presented here is that the methodology has been designed to extend the ID model from (Cobb, 2007). There are other methods that could be employed to determine decision rules for continuous variables with multiple continuous parents, such as a straightforward grid search or a sampling technique. Future research will be aimed at exploring these methods and comparing them with those in this paper.

Acknowledgments

The author is grateful for the insightful comments of three anonymous reviewers. Support from the Spanish Ministry of Science and Innovation through project TIN2007-67418-C03-02 and by EFDR funds is gratefully acknowledged.

References

Cobb, B.R., 2007. Influence diagrams with continuous decision variables and non-Gaussian uncertainties. *Decision Anal.* **4**(3) 136–155.

Cobb, B.R., 2009. Efficiency of influence diagram models with continuous decision variables. *Decis. Support Syst.* **48** 257–266.

Cobb, B.R. 2010. Observing Multiple Continuous Sources of Information in Decision Problems. Working Paper, Virginia Military Institute, Lexington, VA. Available for download at: `www.vmi.edu/fswebs.aspx?tid=24697&id=24791`

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.

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

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.

Generalized Continuous Time Bayesian Networks and their GSPN Semantics

Daniele Codetta-Raiteri, Luigi Portinale
Dipartimento di Informatica, Università del Piemonte Orientale "A. Avogadro"
{dcr, portinal}@di.unipmn.it

Abstract

We present an extension to Continuous Time Bayesian Networks (*CTBN*) called Generalized *CTBN* (*GCTBN*). The formalism allows one to model continuous time delayed variables (with exponentially distributed transition rates), as well as non delayed or "immediate" variables, which act as standard chance nodes in a Bayesian Network. The usefulness of this kind of model is discussed through an example concerning the reliability of a simple component-based system. The interpretation of *GCTBN* is proposed in terms of Generalized Stochastic Petri Nets (*GSPN*); the purpose is twofold: to provide a well-defined semantics for *GCTBN*in terms of the underlying stochastic process, and to provide an actual mean to perform inference (both prediction and smoothing) on *GCTBN*.

1 Introduction

The goal of this paper is to propose a generalization of Continuous Time Bayesian Networks (*CTBN*) (Nodelman et al., 2005) by allowing the presence of nodes with no explicit temporal evolution, called "immediate nodes". The resulting framework is called Generalized *CTBN* (*GCTBN*) and allows the modeling of processes having both a continuous-time temporal component and an immediate component capturing the logical/probabilistic interactions among modeled variables.

The possibilities offered by this generalization, can be exploited in several applications. For example, in system reliability analysis, it is very practical to distinguish between system components (having a temporal evolution) and specific modules or subsystems, whose behavior has to be modeled for the analysis. For instance, in Fault Tree Analysis (Dugan et al., 1992), basic events represent the system components with their failure rates, while non-basic events are logical gates identifying modules of the system under examination. In Dynamic Fault Trees (Dugan et al., 1992), logical gates identifying sub-modules, can be combined with dynamic gates, modeling time-dependent dependencies (usually assuming continuous time) among components or sub-modules. Also in this case, it is very important to distinguish, at the modeling level, between delayed and immediate entities. Of course, similar considerations apply in other tasks as well, as in medical diagnosis, financial forecasting, biological process modeling, etc.

The paper is organized as follows: Sec. 2 provides basic notions about the formalisms involved in the *GCTBN* definition and analysis; in Sec. 3, the *GCTBN* formalism is defined; in Sec. 4, a reliability case study is introduced, together with the corresponding *GCTBN* modeling; in Sec. 5, a semantic model, based on the formalism of Generalized Stochastic Petri Nets (*GSPN*) (Ajmone et al., 1995) is defined, and the corresponding model for the case study is discussed; in Sec. 6, we provide the algorithms to perform inference on a *GCTBN*, by means of analysis on the corresponding *GSPN*.

2 Preliminary notions

CTBN. Probabilistic graphical models for reasoning about processes that evolve over time, allow for a *factorization* (Lauritzen and Richardson, 2002) of the state space of the process, resulting in better modeling and inference features. Such models are usually based on graph structures, grounded on the theory of Bayesian Networks (*BN*). When time is taken into account, the main choice concerns whether to consider it as a discrete or a continuous dimension. In the second case, Continuous Time

Pp. 105–113 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

Bayesian Networks (*CTBN*) have been firstly proposed in (Nodelman et al., 2002; Nodelman et al., 2005) and then refined in (Saria et al., 2007).

Standard inference tasks in temporal probabilistic models are *prediction* and *smoothing*. *Prediction* consists in computing the probability of a future state, given past evidence (a special case occurs when the last evidence time point and the query time are the same and is called *Filtering* or *Monitoring*). *Smoothing* is the task of estimating a past state, given all the evidence (observations) up to now. Such tasks can be accomplished by inference procedures usually based on specific adaptation of standard *BN* algorithms. In case of a *CTBN*, exact inference may often be impractical, so approximations through message-passing algorithms on cluster graphs (Nodelman et al., 2005; Saria et al., 2007), or through sampling (El-Hay et al., 2008; Fan and Shelton, 2008), have been proposed.

CTMC. A Continuous Time Markov Chain (Ajmone et al., 1995) enumerates the possible system states (nodes) and state transitions (arcs). A transition is not immediate, but may occur after a random period of time ruled by the negative exponential distribution according to the transition rate. Besides transition rates, a *CTMC* is characterized by the initial probability distribution of its states. There are two main analyses that can be performed with a *CTMC*: *steady state* and *transient* analysis. In the first case, the equilibrium distribution (at infinite time) of the states is computed, while in the second case, such a distribution is computed at a given time point.

GSPN are a particular form of Petri Nets, so they are composed by places, transitions and arcs (Fig. 2). A place can contain a discrete number of tokens (place *marking*), and the current state of the system is represented by the net marking given by the number of tokens in each place of the net. Transitions are used to model the system state transitions; a transition is enabled to fire when a certain net marking holds, and when the transition fires, a certain amount of tokens is moved from a set of places to another one, changing the net marking, so the system state.

Directed arcs are used to connect places to transitions and vice-versa, with the aim of moving tokens when transitions fire. In *GSPN*, inhibitor arcs are also present and connect a place to a transition with the aim of disabling the transition if the place is not empty. A cardinality can be associated with an arc in order to specify the number of tokens to be moved, in case of directed arcs, or the number of tokens necessary to disable the transition, in case of inhibitor arcs.

In *GSPN*, transitions can be immediate or timed. Immediate transitions fire as soon as they are enabled. In case of concurrent immediate transitions, their firing can be ruled by means of weights or priorities (π). Timed transitions fire if enabled, after a random period of time ruled by the negative exponential distribution according to the firing rate. Vanishing markings are those enabling immediate transitions; tangible markings are those enabling timed transitions.

The stochastic process associated with a *GSPN* is a homogeneous continuous time semi-Markov process (Ajmone et al., 1995) that can be analyzed by removing from the set of possible markings (states), the vanishing markings (since the system does not spend time in such states), and by analyzing the resulting *CTMC*. In this way, the analysis of a *GSPN* can provide several measures, and in particular the transient or steady state probability distribution of the number of tokens in each place.

A *GSPN* model can be edited and analyzed (or simulated) by means of *GreatSPN* (Chiola et al., 1995); in particular, this tool allows to set marking dependent firing rates. This means that the value of the firing rate of a timed transition, can change according to a set of conditions concerning the current marking of specific places. This possibility is exploited to simplify the generation of the *GSPN* model from the *GCTBN* (Portinale and Codetta, 2009). However, a timed transition characterized by a marking dependent firing rate, is equivalent to a

set of timed transitions, each characterized by a certain constant firing rate and enabled by the corresponding condition about the marking of places. The two solutions determine the same underlying *CTMC*.

3 Generalized *CTBN*

Following (Nodelman et al., 2002), a *CTBN* is defined as follows:

Let $X = X_1, \dots X_n$ be a set of discrete variables, a *CTBN* over X consists of two components. The first one is an initial distribution P_X^0 over X (possibly specified as a standard *BN* over X). The second component is a continuous-time transition model specified as (1) a directed graph G whose nodes are $X_1, \dots X_n$ (and with $Pa(X_i)$ denoting the parents of X_i in G); (2) a conditional intensity matrix (*CIM*) $Q_{X_i|Pa(X_i)}$ for every $X_i \in X$.

A *GCTBN* is defined as follows:
Given a set of discrete variables $X = \{X_1, \dots X_n\}$ partitioned into the sets D (delayed variables) and I (immediate variables) (i.e. $X = D \cup I$ and $D \cap I = \emptyset$), a *GCTBN* is a pair $N = \langle P_D^0, G \rangle$ where
-P_D^0 is an initial probability distribution over D;
-G is a directed graph whose nodes are $X_1, \dots X_n$ (and with $Pa(X_i)$ denoting the parents of X_i in G) such that
1. there is no directed cycle in G composed only by nodes in the set I;
2. for each node $I_j \in I$ a conditional probability table (*CPT*) $P[I_j|Pa(I_j)]$ is defined (as in standard *BN*);
3. for each node $D_k \in D$ a *CIM* $Q_{D_k|Pa(D_k)}$ is defined (as in standard *CTBN*).

Delayed nodes are, as in case of a *CTBN*, nodes representing discrete variables with a continuous time evolution: the transition from a value to another one, is ruled by exponential transition rates defined in the *CIM* associated with the node. A delayed node is characterized also by the initial probability distribution of its possible values. So, a delayed node implicitly incorporates a *CTMC* (Sec. 2). If a delayed node is a root node (has no parent nodes) the transition rates are constant, otherwise the rates are conditioned by the values of the parent nodes. Such nodes, in the case of *GCTBN*, may be either delayed or immediate.

Immediate nodes are introduced in order to capture variables whose evolution is not ruled by transition rates associated with their values, but is conditionally and immediately determined, at a given time point, by the values of other variables in the model. Immediate nodes are then treated as usual chance nodes in a *BN* and have a standard *CPT* associated with them. In case of an immediate root node, its *CPT* actually specifies a prior probability distribution.

In a *GCTBN*, an immediate node I_j directly depends on its parent nodes ($Pa(I_j)$). However, if the set $Pa(I_j)$ contains immediate nodes, then the change of such nodes is ruled in turns by the change of their parents, eventually being delayed variables. So, what really determines a change in I_j is not $Pa(I_j)$, but instead the set of the "Closest" Delayed Ancestors of I_j ($CDA(I_j)$). Such set contains any delayed variable D_k such that a path from D_k to I_j exists and contains no intermediate delayed nodes.

The initial distribution P_D^0 is specified only on delayed variables, since this is sufficient to obtain the joint initial distribution over the set X of all the variables of the *GCTBN* as follows: $P_X^0 = P_D^0 \prod_{I_j \in I} P[I_j|Pa(I_j)]$.

A few words are worth to be spent for the structure of the graph modeling the *GCTBN*. While it is in general possible to have cycles in the graph (as in *CTBN*) due to the temporal nature of some nodes, such cycles cannot be composed only by immediate nodes. Indeed, if this would be the case, we would introduce static circular dependencies among model variables.

The evolution of a system modeled through a *GCTBN* occurs as follows: the initial state is given by the assignment of the initial values of the variables, according to P_X^0 (immediate root nodes, if any, keep their initial value during the model evolution). Given the current system state (represented by the joint assignment of the model variables, both delayed and immediate), a value transition of a delayed variable D_k will occur, after an exponentially distributed delay, by producing a new state called a "van-

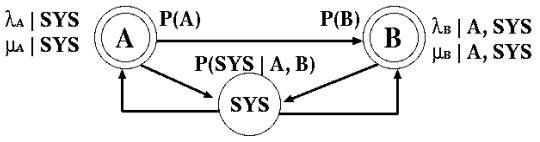


Figure 1: *GCTBN* model of the case study.

ishing state"; given the new vanishing state, a new assignment is determined to any immediate variable I_j such that D_k belongs to $CDA(I_j)$. The assignment to I_j is consistent with the *CPT* of I_j. The resulting state, called a "tangible state", is the new actual state of the system, from which the evolution can proceed, with a new transition of value, by a delayed variable. As we noticed in Sec. 2, the same state classification can be recognized in *GSPN*.

4 A motivating case study

To highlight usefulness and features of a *GCTBN* model, we now consider a simple case study in the field of reliability analysis. It consists of a small system composed by the main component A and its spare component B. Initially A is active while B is dormant; in case of failure of A, B is activated in order to replace A. However, the activation of B may fail with probability 0.01. If B fails before A, B can not replace A. The system is considered as failed if A is failed and B is dormant or failed. We suppose that only while the system is failed, the components A and B undergo repair. As soon as the repair of one of the components is completed, the component re-starts in working state and consequently the system becomes operative again; this implies that the repair of the other component is suspended.

The time to failure of the components is a random variable ruled by the negative exponential distribution: in the case of A, the failure rate is λ_A =1.0E-06 h^{-1}. The failure rate of B, λ_B, changes according to its current state: if B is dormant, λ_B is equal to 5.0E-07 h^{-1}; if instead B is active, λ_B is equal to 1.0E-06 h^{-1}. The time to repair a component is still ruled by the negative exponential distribution: A and B have the same repair rate $\mu_A = \mu_B = 0.01\ h^{-1}$.

a)

1 → 2		2 → 1	
SYS	λ_A	SYS	μ_A
1	1.0E-06 h^{-1}	1	0 h^{-1}
2	1.0E-06 h^{-1}	2	0.01 h^{-1}

b)

1 → 2			2 → 1		
A	SYS	λ_B	A	SYS	μ_B
1	1	5.0E-07 h^{-1}	1	1	0 h^{-1}
1	2	–	1	2	–
2	1	1.0E-06 h^{-1}	2	1	0 h^{-1}
2	2	5.0E-07 h^{-1}	2	2	0.01 h^{-1}

c)

A	B	SYS	Prob.	A	B	SYS	Prob.
1	1	1	1	2	1	1	0.99
1	1	2	0	2	1	2	0.01
1	2	1	1	2	2	1	0
1	2	2	0	2	2	2	1

Table 1: a) *CIM* of A. b) *CIM* of B. c) *CPT* of SYS.

The *GCTBN* model. The case study described above is represented by the *GCTBN* model in Fig. 1 where the variables A, B, SYS represent the state of the components and of the whole system respectively. All the variables are binary because each entity can be in the working state (1) or in the failed state (2); for the component B, the working state comprises both the dormancy and the activation.

The variable A influences the variable B because the failure rate of the component B depends on the state of A. Both the variables A and B influence SYS because the state of the whole system depends on the state of the components A and B. The arcs connecting the variable SYS to A and B respectively, concern the repair of the components A and B only while the system is failed.

A and B are delayed variables (Sec. 3) and implicitly incorporate a *CTMC* composed by two states: 1 and 2. Since both components are initially supposed to work, the initial probability distribution is set equal to 1 for states $A = 1$ and $B = 1$. In the *CIM* of A (Tab. 1.a), we can notice that the rate μ_A is not null only if the value of SYS is 2. The rate λ_A instead, is constant. In the *CIM* of B (Tab. 1.b), λ_B is increased only when A is equal to 2 and SYS is equal to 1 (this implies that B is active). As in the case of the variable A, the rate μ_B is not null only if the value of SYS is 2. The combination $A = 1, SYS = 2$ is impossible, so the corresponding entries are not significant.

The variable SYS is immediate (Sec. 3) and is

characterized by the *CPT* appearing in Tab. 1.c. In particular, SYS is surely equal to 1 if A is equal to 1, and surely equal to 2 if both A and B are equal to 2. In the case of A equal to 2 and B equal to 1, SYS assumes the value 1 with probability 0.99 (this implies the activation of the spare component B), or the value 2 with probability 0.01 (this implies that B fails to activate).

The introduction of the immediate variable SYS is actually an important modeling feature, since it allows one to directly capture the static (or immediate) interactions between A and B, resulting in a probabilistic choice about the whole system status. Without the use of an immediate variable, it is hard to factorize the model using variables A and B (see (Portinale and Codetta, 2009) for an example, where an ordinary *CTBN* may fail in modeling all possible state transitions).

5 A Petri Net semantics for *GCTBN*

Combining in a single model entities explicitly evolving over time with entities whose determination is "immediate", has been already proposed in frameworks other than *CTBN*. In case of continuous time, a model having such features can be found in the framework of Petri Nets, namely *GSPN* (Sec. 2). A *GCTBN* model can be expressed in terms of a *GSPN*, by means of a set of translation rules (see (Portinale and Codetta, 2009) for details). This translation is twofold: (1) it provides a well-defined semantics for a *GCTBN* model, in terms of the underlying stochastic process it represents; (2) it provides an actual mean to perform inference on the *GCTBN* model, by exploiting well-studied analysis techniques for *GSPN*.

In fact, solution techniques for *GSPN* have received a lot of attention, especially with respect to the possibility of representing in a compact way the underlying *CTMC* and in solving it efficiently (Miner, 2007). Once a *GCTBN* has been compiled into a *GSPN*, such techniques can be employed to compute inference measures on the original *GCTBN* model (Sec. 6).

Case study. According to the conversion rules described in (Portinale and Codetta, 2009), the *GCTBN* of the case study in Fig. 1 can be converted into the *GSPN* model shown in Fig. 2 where the places A, B and SYS represent the variables of the *GCTBN* model. The value of a *GCTBN* variable is mapped into the marking (number of tokens) of the corresponding place in the *GSPN*. Let us consider the place B in the *GSPN*: the marking of the place B can be equal to 1 or 2, the same values that the variable B in the *GCTBN* can assume. B is a delayed variable and its initialization is modeled in the *GSPN* by the immediate transitions B_init_1 and B_init_2. Their effect is to set the initial marking of the place B to 1 or 2 respectively. Their weights correspond to the initial probability distribution of the variable B.

The change of the marking of the place B is determined by the timed transitions B_1_2 and B_2_1. The transition B_1_2 is enabled to fire when the place B contains one token; the firing sets the marking of B to 2. The transition B_2_1 instead, can fire when the marking of the place B is equal to 2, and turns it to 1.

The dependency of the transition rate of a variable on the values of the other variables in the *GCTBN* model, becomes in the *GSPN* model, the dependency of the firing rate of a timed transition on the markings of the other places. For instance, in the *GCTBN* model, the variable B depends on A and SYS; in the *GSPN* model, λ_B becomes the firing rate of the timed transition B_1_2, its value depends on the marking of the places A and SYS, and assumes the same values reported in Tab. 1.b. The firing rate of the timed transition B_2_1 instead, is μ_B reported in Tab. 1.b, still depending on the marking of the places A and SYS.

In the *GCTBN*, the variable SYS is immediate and depends on A and B. Therefore in the *GSPN*, each time the marking of the place A or B is modified, the marking of SYS has to be immediately updated: each time the transition A_1_2, A_2_1, B_1_2 or B_2_1 fires, one token appears in the place $emptySYS$; this determines the firing of the immediate transition $reset\_SYS\_1$ or $reset\_SYS\_2$ removing

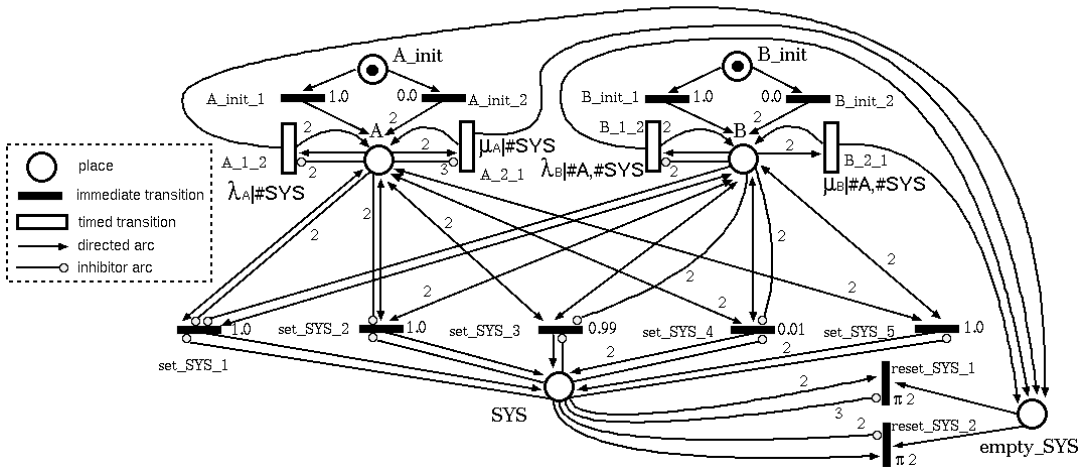


Figure 2: *GSPN* model obtained from the *GCTBN* in Fig. 1.

any token in SYS. Then, the marking of such place is set by one of the immediate transitions set_SYS_1, set_SYS_2, set_SYS_3, set_SYS_4, set_SYS_5. Each of them corresponds to one entry having not null probability in the *CPT* of Tab. 1.c.

Infinite rates. Another advantage of the *GSPN* semantics for *GCTBN* is that we can also model immediate changes on delayed variables; since delayed variables are characterized by changing rates (conditioned on their parents' values), this can be theoretically modeled by "infinite" rate values, but this is unmanageable with standard stochastic analysis. On the other hand, modeling the changes of delayed variables through transitions of a *GSPN* allows one to use immediate transitions to represent the above situation. A practical example can again be found in reliability applications, when the failure of a system component triggers the instantaneous failure of a dependent component (this is usually called a *functional dependency* (Dugan et al., 1992)). Fig. 3 is an example of a functional dependency between a component F (the trigger) and a component A: the failure of F immediately induces a failure of A. Fig. 3 shows the *GCTBN* model and the corresponding *GSPN*.

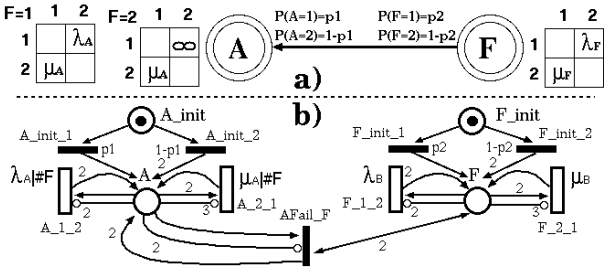


Figure 3: a) *GCTBN* modeling a functional dependency. b) The corresponding *GSPN* model.

6 Inference algorithms

In the present work, we take advantage of the correspondence between *GCTBN* and *GSPN*, in order to propose inference algorithms based on *GSPN* solution algorithms. For instance, computing the probability of a given *GCTBN* variable assignment $X = x_i$ at time t, will correspond to compute the probability of having i tokens at time t in the place modeling X in the *GSPN*.

Standard inference tasks are prediction and smoothing. The prediction task consists in computing $P(Q_t|e_{t_1}, \ldots, e_{t_k})$ which is the posterior probability at time t of a set of queried variables $Q \subseteq (D \cup I)$, given a stream of observations (evidence) $e_{t_1}, \ldots, e_{t_k}$ from time t_1 to time t_k with $t_1 < \ldots t_k < t$. Every evidence e_{t_j} consists of a

Procedure PREDICTION
INPUT: a set of queried variables Q, a query time t, a set of temporally labeled evidences $e_{t_1}, \ldots e_{t_k}$ with $t_1 < \ldots t_k < t$
OUTPUT: $P(Q_t|e_{t_1}, \ldots e_{t_k})$

- let $t_0 = 0$;
for $i = 1$ **to** k **{**
- solve the *GSPN* transient at time $(t_i - t_{i-1})$;
- compute from transient, $p_i(j) = Pr\{X_j|e_{t_i}\}$ for $X_j \in D \cup R$;
- update the weights of the immediate init transitions of X_j according to $p_i(j)$; **}**
- solve the *GSPN* transient at time $(t - t_k)$;
- compute from transient, $r = Pr\{Q\}$;
- output r;

Figure 4: The prediction inference procedure.

(possibly different) set of instantiated variables. Prediction can then be implemented by repeatedly solving the transient (Sec. 2) of the corresponding *GSPN* at the observation and query times. Of course, any observation will condition the evolution of the model, so the suitable conditioning operations must be performed before a new *GSPN* resolution.

The smoothing task consists in computing $P(Q_t|e_{t_1}, \ldots, e_{t_k})$ which is the probability at time t of a set of queried variables $Q \subseteq (D \cup I)$, given a stream of observations (evidence) $e_{t_1}, \ldots, e_{t_k}$ from time t_1 to time t_k with $t < t_1 < \ldots t_k$. The issue is how to condition on variables observed at a time instant that follows the current one. The idea is then to try to reformulate the problem in such a way that it can be reduced to a prediction-like task. The approach is then based on the application of the Bayes rule as follows:
$P(Q_t|e_{t_1}, \ldots e_{t_k}) = \alpha P(Q_t)P(e_{t_1}, \ldots e_{t_k}|Q_t) =$
$= \alpha P(Q_t)P(e_{t_1}|Q_t) \ldots P(e_{t_k}|e_{t_1}, \ldots e_{t_{k-1}}, Q_t)$
In this way, every factor in the above formula is conditioned on the past and can be implemented as in prediction. However, this solution requires the computation of a normalization factor (α).

The pseudo-code for the prediction and smoothing procedure is shown in Fig. 4 and Fig. 5 respectively, and explained in details in (Portinale and Codetta, 2009).

Case study. Consider again the case study of Fig. 1. We can easily compute the unreliability of the whole system, by asking for the probability P($SYS = 2$) over time. This reduces to compute the probability of having 2 tokens into

Procedure SMOOTHING
INPUT: a set of queried variables Q, a query time t, a set of temporally labeled evidences $e_{t_1}, \ldots e_{t_k}$ with $t < t_1 < \ldots t_k$
OUTPUT: $P(Q_t|e_{t_1}, \ldots e_{t_k})$;

{ - Let N be the cardinality of possible assignments $q_i (1 \leq i \leq N)$ of Q;
- A: **array**[N];
for $i = 1$ **to** N A[i]=SMOOTH(q_i); //possibly in parallel
- output **normalize**(A); }

Procedure SMOOTH**(q)** {
- $t_0 = t$;
- solve the *GSPN* transient at time t;
- compute from transient, $r = Pr\{Q = q\}$;
- $ev = q$;
for $i = 1$ **to** k **{**
- compute from transient, $p_{i-1}(j) = Pr\{X_j|ev\}$ for $X_j \in D \cup R$;
- update the weights of the immediate init transitions of X_j according to $p_{i-1}(j)$;
- solve the *GSPN* transient at time $(t_i - t_{i-1})$;
- compute from transient, $p_i(e) = Pr\{e_{t_i}\}$
- $r = r \cdot p_i(e)$;
- $ev = e_{t_i}$; **}**
- output r; }

Figure 5: The smoothing inference procedure.

place SYS, on the corresponding *GSPN*. This is done, by solving the transient (Sec. 2) at the required time instants. Results for our example are reported in Tab. 2. Since the modeled system is repairable, it makes sense to ask for the steady state distribution, in order to understand whether the system is reliable in the long run. By solving the *GSPN* for steady state (Sec. 2), we can indeed compute that the probabilities of component A and B being faulty in the long run are 0.496681 and 0.500026 respectively, while the probability of the whole system being faulty ($P(SYS = 2)$) is 0.000051, meaning that a good reliability is assured.

Concerning prediction, let us consider to observe the system working ($SYS = 1$) at time $t_1 = 10^5 h$ and the system failed ($SYS = 2$) at time $t_2 = 2 \cdot 10^5 h$. As described in details in (Portinale and Codetta, 2009), by applying the procedure outlined in Fig. 4, we can compute the probability of component A being working at time $t = 5 \cdot 10^5 h$, conditioned by the observation stream. The result is 0.521855.

Concerning smoothing inference, let us suppose to have observed the system working at time $t_1 = 3 \cdot 10^5 h$ and failed at time $t_2 = 5 \cdot 10^5 h$. We ask for the probability of component A being failed at time $t = 2 \cdot 10^2 h$, conditioned by the above evidence. By applying the procedure out-

Time (h)	Unreliability	Time (h)	Unreliability
200000	$1.4E-05$	400000	$2.3E-05$
300000	$1.9E-05$	500000	$2.7E-05$

Table 2: Unreliability results.

lined in Fig. 5, as described in details in (Portinale and Codetta, 2009), we obtain that the required probability is equal to 0.308548.

7 Conclusions and future works

The presented formalism of *GCTBN* allows one to mix in the same model continuous time delayed variables with standard "immediate" chance variables, as well as to model immediate changes on delayed variables. The usefulness of this kind of model has been discussed through some examples from reliability analysis.

The semantics of the proposed *GCTBN* formalism has been provided in terms of *GSPN*, a well-known formalism with well established analysis techniques. In particular, adopting *GSPN* solution algorithms as the basis for *GCTBN* inference, allows one to take advantage of specialized methodologies for solving the underlying stochastic process, that are currently able to deal with extremely large models; in particular, such techniques (based on symbolic data structures) allow for one order of magnitude of increase in the size of the models to be solved exactly, with respect to standard methods, meaning that models with an order of 10^{10} tangible states can actually be solved (Miner, 2007).

However, the analysis of a *GCTBN* by means of the underlying *GSPN* is only one possibility that does not take explicit advantage of the structure of the graph as in *CTBN* algorithms (Nodelman et al., 2005; Saria et al., 2007). Our future works will concentrate on the possibility of adopting cluster-based or stochastic simulation approximations, even on *GCTBN* models, and in comparing their performance and quality with respect to *GSPN*-based solution techniques. Finally, since symbolic representations have been proved very useful for the analysis of *GSPN* models, it would also be of significant interest to study the relationships between such representations and the inference procedures on probabilistic graphical models in general, since this could in principle open the possibility of new classes of algorithms for *BN*-based formalisms.

References

M. Ajmone, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. 1995. *Modelling with Generalized Stochastic Petri Nets*. J. Wiley.

G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. 1995. GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation*, 24(1&2):47–68.

J.B. Dugan, S.J. Bavuso, and M.A. Boyd. 1992. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41:363–377.

T. El-Hay, N. Friedman, and R. Kupferman. 2008. Gibbs sampling in factorized continuous time Markov processes. In *Proc. 24rd UAI'08*.

Y. Fan and C. Shelton. 2008. Sampling for approximate inference in continuous time Bayesian networks. In *Proc. 10th Int. Symp. on AI and Mathematics*.

S.L. Lauritzen and T.S. Richardson. 2002. Chain graph models and their causal interpretations. *Journal Of The Royal Statistical Society Series B*, 64(3):321–348.

A.S. Miner. 2007. Decision diagrams for the exact solution of Markov models. *Proceedings in Applied Mathematics and Mechanics (PAMM)*, 7(1).

U. Nodelman, C.R. Shelton, and D. Koller. 2002. Continuous Time Bayesian Networks. In *Proc. 18th UAI'02*, pages 378–387.

U. Nodelman, C.R. Shelton, and D. Koller. 2005. Expectation propagation for continuous time Bayesian networks. In *Proc. 21st UAI'05*, pages 431–440.

L. Portinale and D. Codetta. 2009. A GSPN semantics for continuous time Bayesian networks with immediate nodes. Technical Report TR-INF-2009-03-03-UNIPMN, Dip. di Informatica, Univ. del Piemonte Orientale. http://www.di.unipmn.it.

S. Saria, U. Nodelman, and D. Koller. 2007. Reasoning at the right time granularity. In *Proc. 23rd UAI'07*, pages 421–430.

Same-Decision Probability: A Confidence Measure for Threshold-Based Decisions under Noisy Sensors

Adnan Darwiche and Arthur Choi
Computer Science Department
University of California, Los Angeles
{*darwiche, aychoi*}*@cs.ucla.edu*

Abstract

We consider in this paper the robustness of decisions based on probabilistic thresholds under noisy sensor readings. In particular, we consider the stability of these decisions under different assumptions about the causal mechanisms that govern the output of a sensor. To this effect, we propose the *same-decision probability* as a query that can be used as a confidence measure for threshold-based decisions, and study some of its properties.

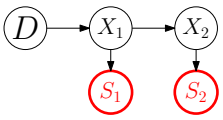
1 Introduction

There has been an increased interest recently in providing assurances on the results of probabilistic reasoning systems. One clear example are the many results on sensitivity analysis, which is concerned with providing guarantees on the relationship between probabilistic queries and model parameters; see, e.g., (Chan, 2009; Kwisthout and van der Gaag, 2008). These results include specific bounds on the changes in probabilistic queries that could result from perturbing model parameters.

We consider another class of assurances in this paper, which is concerned with quantifying the robustness of threshold-based decisions against noisy observations, where we propose a specific notion, called the *same-decision probability*. Our proposed notion is cast in the context of Bayesian networks where the goal is to make a decision based on whether some probability $Pr(d \mid \mathbf{s})$ passes a given threshold T, where $\mathbf{s}$ represents the readings of noisy sensors. The same-decision probability is based on the following key observation. If one were to know the specific causal mechanisms $\mathbf{h}$ that govern the behavior of each noisy sensor (and hence, allow us to precisely interpret each sensor reading), then one should clearly make the decision based on the probability $Pr(d \mid \mathbf{s}, \mathbf{h})$ instead of the probability $Pr(d \mid \mathbf{s})$. In fact, the probability $Pr(d \mid \mathbf{s})$ can be seen as simply the expectation of $Pr(d \mid \mathbf{s}, \mathbf{h})$ with respect to the distribution $Pr(\mathbf{h} \mid \mathbf{s})$ over causal mechanisms. The same-decision probability is then the probability that we would have made the same threshold-based decision had we known the specific causal mechanism $\mathbf{h}$. More precisely, it is the expected decision based on $Pr(d \mid \mathbf{s}, \mathbf{h})$, with respect to the distribution $Pr(\mathbf{h} \mid \mathbf{s})$ over sensor causal mechanisms.

We show a number of results about this proposed quantity. First, we show that a standard Bayesian network does not contain all of the information necessary to pinpoint the distribution $Pr(\mathbf{h} \mid \mathbf{s})$ which is needed for completely defining the same decision probability. We formulate, however, two assumptions, each of which is sufficient to induce this distribution. Second, we propose a bound on the same-decision probability using the one-sided Chebyshev inequality, which requires only the variance of $Pr(d \mid \mathbf{s}, \mathbf{h})$ with respect to the distribution $Pr(\mathbf{h} \mid \mathbf{s})$. Third, we propose a variable elimination algorithm that computes this variance in time and space that are exponential only in the constrained treewidth of the given network. We conclude with a number of concrete examples that illustrate the utility of our proposed confidence measure in quantifying the robustness of threshold-based decisions.

Pp. 113–121 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.



D	$Pr(D\|S_1{=}+, S_2{=}+)$
+	0.880952
−	0.119048

D	$Pr(D)$
+	0.5
−	0.5

D	X_1	$Pr(X_1\|D)$
+	+	0.9
+	−	0.1
−	+	0.1
−	−	0.9

X_i	S_i	$Pr(S_i\|X_i)$
+	+	0.9
+	−	0.1
−	+	0.1
−	−	0.9

Figure 1: A simple Bayesian network, under sensor readings $\{S_1{=}+, S_2{=}+\}$. Variables S_1 and S_2 represent noisy sensor readings, and they have the same CPT $Pr(S_i|X_i)$. Variables X_1 and X_2 also have the same CPTs (only the one for variable X_1 is shown).

Table 1: Causal mechanisms for sensor readings $\mathbf{s} = \{S_1{=}+, S_2{=}+\}$ for the network in Fig. 1. Cases above threshold $T = 0.6$ are in bold.

$\mathbf{h}$	H_1	H_2	$Pr(\mathbf{h}\|s_1, s_2)$	$Pr(d\|s_1, s_2, \mathbf{h})$
1	t	t	0.781071	**0.90**
2	p	t	0.096429	**0.82**
3	l	t	0.001071	0.10
4	t	p	0.096429	**0.90**
5	p	p	0.021429	0.50
6	l	p	0.001190	0.10
7	t	l	0.001071	**0.90**
8	p	l	0.001190	0.18
9	l	l	0.000119	0.10

2 An Introductory Example

Consider the Bayesian network in Figure 1, which models a scenario involving a hypothesis D of interest and two noisy sensors S_1 and S_2 that bear on this hypothesis. The network captures the false positive and false negative rates of these sensors, where each sensor S_i is meant to measure the state of variable X_i. A typical usage of this and similar networks involves the computation of our belief in the hypothesis given some sensor readings, $Pr(d \mid s_1, s_2)$. This belief can then be the basis of a decision that depends on whether this probability exceeds a certain threshold, $Pr(d \mid s_1, s_2) \geq T$. Scenarios such as this are typical in applications such as diagnosis (Hamscher et al., 1992), troubleshooting (Heckerman et al., 1995a), and probabilistic planning (Littman et al., 1998).

Figure 1 shows a particular reading of two sensors and the resulting belief $Pr(D{=}+ \mid S_1{=}+, S_2{=}+)$. If our threshold is $T = 0.6$, then our computed belief confirms the decision under consideration. This decision, however, is based on the readings of two noisy sensors. Suppose now that our model had explicated the causal mechanisms that led to the sensor readings we observed, as depicted in Table 1 (we discuss how to obtain such a model in the next section). This table depicts a distribution over causal mechanisms $Pr(\mathbf{h} \mid s_1, s_2)$. Assuming a particular causal mechanism $\mathbf{h}$ is the active one, we also have a refined belief $Pr(d \mid s_1, s_2, \mathbf{h})$ on the hypothesis d. In fact, the original belief $Pr(d \mid s_1, s_2)$ can now be seen as the expectation of the refined beliefs with respect to the distribution over causal mechanisms:

$$Pr(d|s_1, s_2) = \sum_{\mathbf{h}} Pr(d|s_1, s_2, \mathbf{h}) Pr(\mathbf{h}|s_1, s_2).$$

We show that this is the case in general, later.

Suppose now that we knew the specific causal mechanism $\mathbf{h}$ that governs our sensor readings. We would then be able to (and would prefer to) make a decision based on the probability $Pr(d \mid s_1, s_2, \mathbf{h})$ instead of the probability $Pr(d \mid s_1, s_2)$, which again, is only an average over possible mechanisms $\mathbf{h}$. Consider for example Table 1 which enumerates all nine causal mechanisms. In only four of these cases does the probability of the hypothesis pass the given threshold (in bold), leading to the same decision. In the other five scenarios, a different decision would have been made. Clearly, the extent to which this should be of concern will depend on the likelihood of these last five scenarios. As such, we propose to quantify the confidence in our decision using the *same-decision probability:* the probability that we would have made the same decision had we known the causal mechanisms governing a sensor's readings. For this example, this probability is 0.975, indicating a relatively robust decision.

3 Noisy Sensors

In this section, we show how we can augment a sensor so that its causal mechanisms are modeled explicitly. The ultimate goal is to construct

models like the one in Table 1, which are needed for defining the same-decision probability.

Consider a Bayesian network fragment $X \rightarrow S$, where S represents a sensor that bears on variable X and suppose that both S and X take values in $\{+, -\}$.[1] Suppose further that we are given the false positive f_p and false negative f_n rates of the sensor:

$$Pr(S{=}+|X{=}-) = f_p,\ Pr(S{=}-|X{=}+) = f_n.$$

Our augmented sensor model is based on a functional interpretation of the causal relationship between a sensor S and the event X that it bears on. This causal perspective in turn is based on Laplace's conception of natural phenomena (Pearl, 2009, Section 1.4). In particular, we assume that the output of a sensor S is a deterministic function that depends on the state of X, and that the stochastic nature of the sensor arises from the uncertainty in *which* functional relationship manifests itself.

We propose to expand the above sensor model into $X \rightarrow S \leftarrow H$, where variable H is viewed as a selector for one of the four possible Boolean functions mapping X to S, which we ascribe the labels $\{\mathsf{t}, \mathsf{l}, \mathsf{p}, \mathsf{n}\}$:

H	X	S	$Pr(S\|H,X)$
t	+	+	1
t	−	+	0
l	+	+	0
l	−	+	1

H	X	S	$Pr(S\|H,X)$
p	+	+	1
p	−	+	1
n	+	+	0
n	−	+	0

We observe that these Boolean function have commonly used diagnostic interpretations, describing the behavior of a sensor. The state $H{=}\mathsf{t}$ indicates the sensor is truthful, $H{=}\mathsf{l}$ indicates it is lying, $H{=}\mathsf{p}$ indicates it is stuck positive and $H{=}\mathsf{n}$ indicates it is stuck negative. Note that any stochastic model can be emulated by a functional one, with stochastic inputs (Pearl, 2009; Druzdzel and Simon, 1993).

To reason about our augmented sensor model $X \rightarrow S \leftarrow H$, we need to specify a prior distribution $Pr(H)$ over causal mechanisms. Moreover, we need to specify one that yields a model equivalent to the original model $X \rightarrow S$, when variable H has been marginalized out:

$$\begin{aligned} &Pr(S{=}+|X{=}-) \\ &= \textstyle\sum_H Pr(S{=}+|H, X{=}-)Pr(H) = f_p \quad (1) \\ &Pr(S{=}-|X{=}+) \\ &= \textstyle\sum_H Pr(S{=}-|H, X{=}+)Pr(H) = f_n \quad (2) \end{aligned}$$

There is not enough information in the given Bayesian network to identify a unique prior $Pr(H)$. However, if we make some assumptions about this prior, we may be able to pin down a unique one. We make two such proposals here.

For our first proposal, assume that the probability $Pr(H{=}\mathsf{l})$ that a sensor lies is zero, which is a common assumption made in the diagnostic community. This assumption, along with Equations 1 and 2, immediately commits us to the following distribution over causal mechanisms:

H	t	p	n	l
$Pr(H)$	$1 - f_p - f_n$	f_p	f_n	0

For our second proposal, consider the event $\alpha_p = \{H{=}\mathsf{p} \vee H{=}\mathsf{l}\}$ which denotes the materialization of a causal mechanism that produces a false positive behavior by the sensor. That is, if α_p holds, the sensor will report a positive reading when variable X is negative. Moreover, the event $\alpha_n = \{H{=}\mathsf{n} \vee H{=}\mathsf{l}\}$ denotes the materialization of a causal mechanism that produces a false negative behavior by the sensor. Now, if we further assume that the false positive and negative mechanisms of the sensor are independent, we get $Pr(\alpha_p, \alpha_n) = Pr(\alpha_p)Pr(\alpha_n)$. Since α_p, α_n is equivalent to $H{=}\mathsf{l}$, we now get

$$Pr(H{=}\mathsf{l}) = f_p f_n. \quad (3)$$

This assumption, with Equations 1 and 2, commits us to the following CPT:

H	$Pr(H)$
t	$(1 - f_p)(1 - f_n)$
p	$f_p(1 - f_n)$
n	$(1 - f_p)f_n$
l	$f_p f_n$

The assumption is similar to parameter independence used in learning Bayesian networks

[1] Our discussion focuses on sensors over binary variables, but generalizing to multi-valued variables is not difficult; see also (Druzdzel and Simon, 1993).

Table 2: Causal mechanisms for sensor readings $\mathbf{s} = \{S_1{=}+, S_2{=}-\}$ for the network in Fig. 1. Cases above threshold $T = 0.6$ are in bold.

$\mathbf{h}$	H_1	H_2	$Pr(\mathbf{h}\|s_1, s_2)$	$Pr(d\|s_1, s_2, \mathbf{h})$
1	t	t	0.268893	**0.90**
2	p	t	0.298770	0.18
3	l	t	0.029877	0.10
4	t	n	0.298770	**0.90**
5	p	n	0.066393	0.50
6	l	n	0.003689	0.10
7	t	l	0.029877	**0.90**
8	p	l	0.003689	**0.82**
9	l	l	0.000041	0.10

(Heckerman et al., 1995b).[2] Interestingly, under this assumption (and $f_p + f_n < 1$), as the probabilities of $H{=}\mathsf{p}$ and $H{=}\mathsf{n}$ go to zero (i.e., the sensor does not get stuck), the probability of $H{=}\mathsf{l}$ also goes to zero, therefore, implying that the sensor must be truthful.

Note that the two assumptions discussed above become equivalent as the false positive and negative rates of a sensor approach zero. In fact, as we shall illustrate later, the same-decision probability is almost the same when these rates are small, which is the more interesting case. We stress here, however, that the same decision-probability, as a notion, is independent of the specific assumption adopted — and so are the corresponding computational results we shall present later on computing and bounding this probability.

4 Beliefs Based On Noisy Sensors

Suppose now that we have observed the values of n sensors. For a sensor with a positive reading, the three possible states are $\{\mathsf{t}, \mathsf{l}, \mathsf{p}\}$, since the probability $Pr(H{=}\mathsf{n})$ that a sensor is stuck-negative is zero when we have a positive reading. Similarly, for a sensor with a negative reading, the three possible states are $\{\mathsf{t}, \mathsf{l}, \mathsf{n}\}$. Hence, we have (at most) 3^n sensor states that have non-zero probability. Each one of these 3^n states are causal mechanisms, and each refers to a hypothesis about which sensors are truthful, which are lying and which are irrelevant. Table 1 depicts the nine causal mechanisms corresponding to two positive sensor readings in the network of Figure 1. The table also depicts the posterior distribution over these mechanisms, suggesting that the overwhelming leading scenario is the one in which the two sensors are truthful ($\mathbf{h}_1$). Table 2 depicts the nine causal mechanisms assuming two conflicting sensor readings.

Given a reading $\mathbf{s}$ of sensors $\mathbf{S}$, and letting $\mathbf{h}$ range over the causal mechanisms, we now have:

$$\begin{aligned} Pr(d \mid \mathbf{s}) &= \sum_{\mathbf{h}} Pr(d \mid \mathbf{h}, \mathbf{s}) Pr(\mathbf{h} \mid \mathbf{s}) \quad (4) \\ &= \sum_{\mathbf{h}} Q(\mathbf{h}) Pr(\mathbf{h} \mid \mathbf{s}). \end{aligned}$$

We thus view the probability $Pr(d \mid \mathbf{s})$ as an expectation $\mathrm{E}[\, Q(\mathbf{H}) \,]$ with respect to the distribution $Pr(\mathbf{H} \mid \mathbf{s})$ over causal mechanisms, where $Q(\mathbf{h}) = Pr(d \mid \mathbf{h}, \mathbf{s})$.

Table 1 depicts the posterior over causal mechanisms given two positive sensor readings in the network of Figure 1. We have $Pr(D{=}+ \mid S_1{=}+, S_2{=}+) = 0.880952$ in this case, which one can easily verify as also being the expectation of $Pr(D{=}+ \mid S_1{=}+, S_2{=}+, \mathbf{h})$ with respect to the distribution $Pr(\mathbf{h} \mid S_1{=}+, S_2{=}+)$. Table 2 depicts another posterior over causal mechanisms given two conflicting sensor readings. We have $Pr(D{=}+ \mid S_1{=}+, S_2{=}-) = 0.631147$ in this case.

5 Same-Decision Probability

As mentioned in the introduction, one is usually interested in making a decision depending on whether the probability of some hypothesis d is no less than some threshold T. Assuming that we know the correct causal mechanism $\mathbf{h}$ governing the sensors readings $\mathbf{s}$, we clearly want to make this decision based on whether the probability $Q(\mathbf{h}) = Pr(d \mid \mathbf{s}, \mathbf{h})$ is no less than threshold T. However, as we usually do not know the correct causal mechanism, we end up averaging over all such hypotheses, leading to the expectation $Pr(d \mid \mathbf{s})$, and then making a decision depending on whether $Pr(d \mid \mathbf{s}) \geq T$.

[2] Namely, using a Dirichlet prior on the CPT of S in the original model $X \rightarrow S$ would basically assume independent false positive and false negative rates.

Our interest now is in quantifying our confidence in such a decision given that we do not know the correct causal mechanism. Since $Q(\mathbf{H})$ is a random variable, we propose to quantify such a confidence using the following, which we call the *same-decision probability:*

$$\mathcal{P}(Q(\mathbf{H}) \geq T) = \sum_{\mathbf{h}} [Q(\mathbf{h}) \geq T] Pr(\mathbf{h} \mid \mathbf{s}), \quad (5)$$

where $[Q(\mathbf{h}) \geq T]$ is an indicator function that is 1 if $Q(\mathbf{h}) \geq T$ and 0 otherwise. This is the *probability that we would have made the same decision had we known the correct causal mechanisms governing the sensor readings.*

Consider now Equation 5 in relation to Equation 4. Both equations define expectations with respect to the same distribution $Pr(\mathbf{h} \mid \mathbf{s})$. In Equation 4, the resulting expectation is the probability $Pr(d \mid \mathbf{s})$. In Equation 5, the expectation is the same-decision probability. One key difference between the two expectations is that the one in Equation 4 is invariant to the specific distributions used for variables H, as long as these distributions satisfy Equations 1 and 2. However, the expectation in Equation 5 — that is, the same-decision probability — is indeed dependent on the specific distributions used for variables H.

Consider now Table 1, which corresponds to two positive sensor readings in Figure 1. Assuming a threshold of $T = 0.60$, a decision is confirmed given that we have $Pr(D{=}+ \mid S_1{=}+, S_2{=}+) = 0.880952 \geq T$. We make the same decision, however, in only four of the nine causal mechanisms. These probabilities add up to 0.975; hence, the same-decision probability is 0.975. Consider now Table 2, which corresponds to two conflicting sensor readings. The decision is also confirmed here since $Pr(D{=}+ \mid S_1{=}+, S_2{=}-) = 0.631147 \geq T$. Again, we make the same decision in four causal mechanisms, although they are now less likely scenarios. The same-decision probability is only 0.601229, suggesting a smaller confidence in the decision in this case.

Although computing the same-decision probability may be computationally difficult, the one-sided Chebyshev inequality can be used to bound it. According to this inequality, if V is a random variable with expectation $\mathrm{E}[\,V\,] = \mu$ and variance $\mathrm{Var}[\,V\,] = \sigma^2$, then for any $a > 0$:

$$\mathcal{P}(V \geq \mu - a) \geq 1 - \frac{\sigma^2}{\sigma^2 + a^2}$$

Recall now that the probability $Pr(d \mid \mathbf{s})$ is an expectation $\mathrm{E}[\,Q(\mathbf{H})\,]$ with respect to the distribution $Pr(\mathbf{H} \mid \mathbf{s})$, where $Q(\mathbf{h}) = Pr(d \mid \mathbf{h}, \mathbf{s})$. Suppose that $\mathrm{E}[\,Q(\mathbf{H})\,] \geq T$ and a decision has been confirmed accordingly. The same-decision probability is simply the probability of $Q(\mathbf{H}) \geq T$, where $Q(\mathbf{H})$ is a random variable. Using the Chebyshev inequality, we get the following bound on the same-decision probability:

$$\mathcal{P}(Q(\mathbf{H}) \geq T) \geq 1 - \frac{\mathrm{Var}[\,Q(\mathbf{H})\,]}{\mathrm{Var}[\,Q(\mathbf{H})\,] + [Pr(d|\mathbf{s}) - T]^2}$$

Suppose now that $\mathrm{E}[\,Q(\mathbf{H})\,] \leq T$ and a decision has been confirmed accordingly. The same-decision probability in this case is the probability of $Q(\mathbf{H}) \leq T$. Using the Chebyshev inequality now to bound $\mathcal{P}(V \leq \mu + a)$, we get the same bound for the same-decision probability $\mathcal{P}(Q(\mathbf{H}) \leq T)$. To compute these bounds, we need the variance $\mathrm{Var}[\,Q(\mathbf{H})\,]$. We provide an algorithm for this purpose in the next section.

6 Computing the Variance

Let $\mathbf{S}$ and $\mathbf{H}$ be any two disjoint sets of variables in a Bayesian network, with neither set containing variable D. The probability $Pr(d \mid \mathbf{s})$ can be interpreted as an expectation of $Q(\mathbf{h}) = Pr(d \mid \mathbf{s}, \mathbf{h})$ with respect to a distribution $Pr(\mathbf{h} \mid \mathbf{s})$. We propose in this section a general algorithm for computing the variance of such expectations.

Consider now the variance:

$$\begin{aligned}\mathrm{Var}[\,Q(\mathbf{H})\,] &= \mathrm{E}[\,Q(\mathbf{H})^2\,] - \mathrm{E}[\,Q(\mathbf{H})\,]^2 \\ &= \big[\sum_{\mathbf{h}} Pr(d \mid \mathbf{h}, \mathbf{s})^2 Pr(\mathbf{h} \mid \mathbf{s})\big] - Pr(d \mid \mathbf{s})^2\end{aligned}$$

We need two quantities to compute this variance. First, we need the quantity $Pr(d \mid \mathbf{s})$, which can be computed using standard algorithms for Bayesian network inference, such as variable elimination (Zhang and Poole, 1996; Dechter, 1996; Darwiche, 2009). The other

Algorithm 1 Variance by Variable Elimination

input:
$\mathcal{N}$: a Bayes net with distribution Pr
D, d: decision variable and decision state
$\mathbf{S}, \mathbf{s}$: set of sensor variables and readings
$\mathbf{H}$: set of health variables for sensors $\mathbf{S}$

output: a factor that contains $\sum_{\mathbf{h}} \frac{Pr(d,\mathbf{h},\mathbf{s})^2}{Pr(\mathbf{h},\mathbf{s})}$

main:

1: $\mathcal{S}_1 \leftarrow$ factors of $\mathcal{N}$ under observations $d, \mathbf{s}$
2: $\mathcal{S}_2 \leftarrow$ factors of $\mathcal{N}$ under observations $\mathbf{s}$
3: $\mathbf{Y} \leftarrow$ all variables in $\mathcal{N}$ but variables $\mathbf{H}$
4: $\pi \leftarrow$ an ordering of variables $\mathbf{Y}$
5: $\mathcal{S}_1 \leftarrow \text{VE}(\mathcal{S}_1, \mathbf{Y}, \pi)$
6: $\mathcal{S}_2 \leftarrow \text{VE}(\mathcal{S}_2, \mathbf{Y}, \pi)$
7: $\mathcal{S} \leftarrow \{\chi_a \mid \chi_a = \frac{\phi_a^2}{\psi_a}$ for $\phi_a \in \mathcal{S}_1, \psi_a \in \mathcal{S}_2\}$
8: $\pi \leftarrow$ an ordering of variables $\mathbf{H}$
9: $\mathcal{S} \leftarrow \text{VE}(\mathcal{S}, \mathbf{H}, \pi)$
10: **return** $\prod_{\psi \in \mathcal{S}} \psi$

quantity involves a summation over instantiations $\mathbf{h}$. Naively, we could compute this sum by simply enumerating over all instantiations $\mathbf{h}$, using again the variable elimination algorithm to compute the relevant quantities for each instantiation $\mathbf{h}$. However, the number of instantiations $\mathbf{h}$ is exponential in the number of variables in $\mathbf{H}$ and will thus be impractical when the number of such variables is large enough.

However, with a suitably augmented variable elimination algorithm, we can compute this summation more efficiently, and thus the variance. First, consider the following alternative form for the summation:

$$\sum_{\mathbf{h}} Pr(d \mid \mathbf{h}, \mathbf{s})^2 Pr(\mathbf{h} \mid \mathbf{s}) = \frac{1}{Pr(\mathbf{s})} \sum_{\mathbf{h}} \frac{Pr(d,\mathbf{h},\mathbf{s})^2}{Pr(\mathbf{h},\mathbf{s})}.$$

Note that the term $Pr(\mathbf{s})$ is readily available using variable elimination and can be computed together with $Pr(d \mid \mathbf{s})$. Hence, we just need the sum $\sum_{\mathbf{h}} \frac{Pr(d,\mathbf{h},\mathbf{s})^2}{Pr(\mathbf{h},\mathbf{s})}$, which, as we show next, can be computed using an augmented version of variable elimination.[3]

Let $\mathbf{Y}$ denote all variables in the Bayesian network excluding variables $\mathbf{H}$. If we set evidence $\mathbf{s}$ and use variable elimination to sum out variables $\mathbf{Y}$, we get a set of factors that represents the following distribution: $Pr(\mathbf{H}, \mathbf{s}) = \prod_a \psi_a(\mathbf{X}_a)$. Here, ψ_a are the factors remaining from variable elimination after having eliminated variables $\mathbf{Y}$.

We can similarly run the variable elimination algorithm with evidence $\mathbf{s}, d$ to obtain a set of factors whose product represents the following distribution: $Pr(\mathbf{H}, d, \mathbf{s}) = \prod_a \phi_a(\mathbf{X}_a)$. Using the same variable ordering when eliminating variables $\mathbf{Y}$, we can ensure a one-to-one correspondence between factors in both factorizations: each pair of factors ψ_a and ϕ_a will be over the same set of variables $\mathbf{X}_a$ for a given index a. For each instantiation $\mathbf{h}, d, \mathbf{s}$, we have

$$\frac{Pr(\mathbf{h}, d, \mathbf{s})^2}{Pr(\mathbf{h}, \mathbf{s})} = \prod_a \frac{\phi_a(\mathbf{x}_a)^2}{\psi_a(\mathbf{x}_a)},$$

where $\mathbf{x}_a$ is an instantiation of variables $\mathbf{X}_a$ consistent with instantiation $\mathbf{h}, d, \mathbf{s}$. We now compute a new set of factors $\chi_a(\mathbf{X}_a) = \frac{\phi_a(\mathbf{X}_a)}{\psi_a(\mathbf{X}_a)}$ and run the variable elimination algorithm a third time to eliminate variables $\mathbf{H}$ from the factors $\chi_a(\mathbf{X}_a)$. The result will be a trivial factor that contains the quantity of interest.[4]

Algorithm 2 Variable Elimination [VE]

input:
$\mathcal{S}$: set of factors
$\mathbf{Y}$: variables to eliminate in factor set $\mathcal{S}$
π: ordering of variable $\mathbf{Y}$

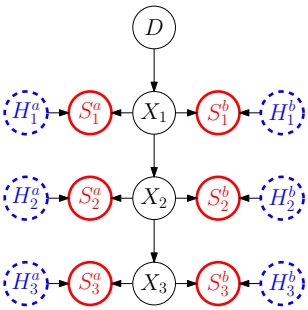
output: factor set where variables $\mathbf{Y}$ are eliminated

main:

1: **for** $i = 1$ to length of order π **do**
2: $\quad \mathcal{S}_i \leftarrow$ factors in $\mathcal{S}$ containing variable $\pi(i)$
3: $\quad \psi_i \leftarrow \sum_{\pi(i)} \prod_{\psi \in \mathcal{S}_i} \psi$
4: $\quad \mathcal{S} \leftarrow \mathcal{S} - \mathcal{S}_i \cup \{\psi_i\}$
5: **return** $\mathcal{S}$

[3] Formally, our summation should be over instantiations $\mathbf{h}$ where $Pr(\mathbf{h}, \mathbf{s}) > 0$. Note that if $Pr(\mathbf{h}, \mathbf{s}) = 0$ then $Pr(d, \mathbf{h}, \mathbf{s}) = 0$. Hence, if we define $x/0 = 0$, then our summation is simply over all instantiations $\mathbf{h}$. In Algorithm 1, we thus define factor division such that $\phi_a(\mathbf{x}_a)^2/\psi_a(\mathbf{x}_a) = 0$ when $\psi_a(\mathbf{x}_a) = 0$.

[4] According to the formulation of variable elimination in (Darwiche, 2009), a trivial factor is a factor over the empty set of variables and contains one entry. It results from eliminating all variables from a set of factors.



D	$Pr(D)$
+	0.25
−	0.75

D	X_1	$Pr(X_1 \mid D)$
+	+	0.9
+	−	0.1
−	+	0.1
−	−	0.9

Figure 2: All sensors have $f_p = f_n = .05$. Variables X_i all have the same CPTs.

Algorithm 1 provides pseudo-code that implements this procedure. Note that on Line 7, there is a one-to-one correspondence between the factors of $\mathcal{S}_1$ and $\mathcal{S}_2$ as we have a one-to-one correspondence between the factors passed to $\text{VE}(\mathcal{S}_1, \mathbf{Y}, \pi)$ and $\text{VE}(\mathcal{S}_2, \mathbf{Y}, \pi)$, and since each call eliminates the same set of variables using the same variable order. Algorithm 1 must eliminate variables $\mathbf{H}$ last, so the complexity of the algorithm is exponential in the *constrained treewidth* (Darwiche, 2009). This is analogous to the complexity of variable elimination for computing MAP, where variables $\mathbf{H}$ are MAP variables (Park and Darwiche, 2004).

7 Examples

Consider the Bayesian network in Figure 2, which depicts a chain D, X_1, X_2, X_3 with two sensors S_i^a and S_i^b attached to each node X_i. Our goal here is to make a decision depending on whether $Pr(D{=}+ \mid \mathbf{s}) \geq T$ for some sensor reading $\mathbf{s}$ and threshold $T = 0.5$. We will next consider a number of sensor readings, each leading to the same decision but a different same-decision probability. Our purpose is to provide concrete examples of this probability, and to show that it can discriminate among sensor readings that not only lead to the same decision, but also under *very similar probabilities* for the hypothesis of interest. The examples will also shed more light on the tightness of the one-sided Chebyshev bound proposed earlier.

Our computations in this section assume the independence between the mechanisms governing false positives and negatives, which is needed to induce a distribution over causal mechanisms (as in Section 3). We also provide the results under the second assumption proposed where the lying causal mechanism has zero probability (in brackets). As we discussed earlier, we expect the two results to be very close since the false positive and negative rates are small, which is confirmed empirically here.

We start by observing that $Pr(D{=}+) = 25\%$. Suppose now that we have a positive reading for sensor S_2^a. We now have the *hypothesis probability* $Pr(D{=}+ \mid S_2^a{=}+) = 55.34\%$ and the decision is confirmed given our threshold. The same-decision probability is 86.19%. From now on, we will say that our *decision confidence* is 86.19% in this case.

The following table depicts what happens when we obtain another positive sensor reading.

	Scenario 1	Scenario 2
sensor readings	$S_2^a{=}+$	$S_2^a{=}+$ $S_2^b{=}+$
hyp. prob.	55.34%	60.01%
dec. conf.	86.19%[85.96%]	99.22%[99.19%]

Note how the decision confidence has increased significantly even though the change in the hypothesis probability is relatively modest. The following table depicts a scenario when we have two more sensor readings that are conflicting.

	Scenario 2	Scenario 3
readings	$S_2^a{=}+$, $S_2^b{=}+$	$S_1^a{=}+$, $S_1^b{=}-$ $S_2^a{=}+$, $S_2^b{=}+$
hyp. prob.	60.01%	60.01%
dec. conf.	99.22%[99.19%]	79.97%[80.07%]

Note how the new readings keep the hypothesis probability the same, but reduce the decision confidence significantly. This is mostly due to raising the probability of some causal mechanism under which we would make a different decision. The following table depicts a conflict between a different pair of sensors.

	Scenario 3	Scenario 4
readings	$S_1^a{=}+$, $S_1^b{=}-$ $S_2^a{=}+$, $S_2^b{=}+$	$S_2^a{=}+$, $S_2^b{=}+$ $S_3^a{=}+$, $S_3^b{=}-$
hyp. prob.	60.01%	60.01%
dec. conf.	79.97%[80.07%]	99.48%[99.48%]

In this case, the sensor conflict increases the same-decision probability slightly (from 99.22%

in Scenario 2 to 99.48%).[5] The next example shows what happens when we get two negative readings but at different sensor locations.

	Scenario 5	Scenario 6
readings	$S_1^a{=}-$, $S_1^b{=}-$ $S_2^a{=}+$, $S_2^b{=}+$	$S_2^a{=}+$, $S_2^b{=}+$ $S_3^a{=}-$, $S_3^b{=}-$
hyp. prob.	4.31%	57.88%
dec. conf.	98.73%[98.70%]	95.25%[95.23%]

When the negative sensors are close to the hypothesis, they reduce the hypothesis probability significantly below the threshold, leading to a high confidence decision. When the readings are further away from the hypothesis (and dominated by the two positive readings), they reduce the hypothesis probability, yet keep it above the threshold. The decision confidence is also reduced, but remains relatively high. Finally, consider the table below which compares the decision confidence, the bound on the confidence, and the variance used to compute the bound.

Scenario	confidence		bound	variance
1	86.19%	$\geq$	15.53%	$1.54 \cdot 10^{-2}$
2	99.22%	$\geq$	90.50%	$1.05 \cdot 10^{-3}$
3	79.97%	$\geq$	11.05%	$8.06 \cdot 10^{-2}$
4	99.48%	$\geq$	88.30%	$1.32 \cdot 10^{-3}$
5	98.73%	$\geq$	98.02%	$4.22 \cdot 10^{-3}$
6	95.25%	$\geq$	34.73%	$1.16 \cdot 10^{-2}$

Note that our decision confidence is high when our bound on the same-decision probability is high. Moreover, the one-sided Chebyshev inequality may provide only weak bounds, which may call for exact computation of the same-decision probability. We computed this quantity through exhaustive enumeration here, yet an algorithm that is exponential only in the constrained treewidth could open new possibilities for reasoning about threshold-based decisions.

8 Conclusion

We considered in this paper the robustness of decisions based on probabilistic thresholds under noisy sensor readings. In particular, we suggested a confidence measure for threshold-based decisions which corresponds to the probability that one would have made the same decision if one had knowledge about a sensor's causal mechanisms. We used the one-sided Chebyshev inequality to bound this probability, which requires computing the variance of a conditional probability with respect to the marginal distribution over a subset of network variables. We also proposed a variable elimination algorithm for computing this variance, whose complexity is exponential only in the constrained treewidth of the given network. We finally provided a number of concrete examples showing the utility of our proposed confidence measure in quantifying the robustness of threshold-based decisions.

[5] Knowing that sensor S_3^b is lying, or that S_3^a is telling the truth, is enough to confirm our decision given our threshold. The conflicting sensor readings thus introduce new scenarios under which the decision is confirmed, although these scenarios are very unlikely.

References

Hei Chan. 2009. *Sensitivity Analysis of Probabilistic Graphical Models: Theoretical Results and Their Applications on Bayesian Network Modeling and Inference*. VDM Verlag.

Adnan Darwiche. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Rina Dechter. 1996. Bucket elimination: A unifying framework for probabilistic inference. In *UAI*, pages 211–219.

Marek J. Druzdzel and Herbert A. Simon. 1993. Causality in Bayesian belief networks. In *UAI*, pages 3–11.

Walter Hamscher, Luca Console, and Johan de Kleer. 1992. *Readings in model-based diagnosis*. MK.

David Heckerman, John S. Breese, and Koos Rommelse. 1995a. Decision-theoretic troubleshooting. *CACM*, 38(3):49–57.

David Heckerman, Dan Geiger, and David Maxwell Chickering. 1995b. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243.

Johan Kwisthout and Linda C. van der Gaag. 2008. The computational complexity of sensitivity analysis and parameter tuning. In *UAI*, pages 349–356.

Michael L. Littman, Judy Goldsmith, and Martin Mundhenk. 1998. The computational complexity of probabilistic planning. *JAIR*, 9:1–36.

James Park and Adnan Darwiche. 2004. Complexity results and approximation strategies for MAP explanations. *JAIR*, 21:101–133.

Judea Pearl. 2009. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition.

Nevin Lianwen Zhang and David Poole. 1996. Exploiting causal independence in Bayesian network inference. *JAIR*, 5:301–328.

On Causal Discovery from Time Series Data using FCI

Doris Entner[1] and Patrik O. Hoyer[1,2]
[1] HIIT & Dept. of Computer Science, University of Helsinki, Finland
[2] CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

We adapt the Fast Causal Inference (FCI) algorithm of Spirtes et al. (2000) to the problem of inferring causal relationships from time series data and evaluate our adaptation and the original FCI algorithm, comparing them to other methods including Granger causality. One advantage of FCI based approaches is the possibility of taking latent confounding variables into account, as opposed to methods based on Granger causality. From simulations we see, however, that while the FCI based approaches are in principle quite powerful for finding causal relationships in time series data, such methods are not very reliable for most practical sample sizes. We further apply the framework to microeconomic data on the dynamics of firm growth. By releasing the full computer code for the method we hope to facilitate the application of the procedure to other domains.

1 Introduction

One of the fundamental goals in empirical science is to discover causal relationships. The most reliable approach towards this goal is performing controlled experiments; regrettably, such experiments are not always possible, for technical or ethical reasons, or high cost. Thus, scientists often seek (preliminary) causal inferences from non-experimental data.

Such data often come in the form of multivariate time series. For instance, economists study the evolution of variables such as GDP, unemployment, and interest rates, while biologists may look at the population dynamics of a set of species. In these cases, a number of variables have been measured repeatedly over time, and the goal is often to understand the myriad ways in which the variables interact.

In time series analysis, most approaches to causal inference are based on Granger causality (Granger, 1969). The basic idea is to, for each variable, identify the set of other variables whose past values are necessary and sufficient for optimal prediction. Unfortunately, if unmeasured variables directly affect two or more of the observed variables, one cannot directly interpret the results of such an analysis.

There exist, however, inference procedures which are asymptotically correct even in the presence of hidden variables. We adapt and evaluate the Fast Causal Inference (FCI) method of Spirtes et al. (2000), originally developed for the analysis of non-temporal variables, to time series data. We demonstrate the statistical behavior of the adapted procedure using numerical simulations, and compare it to other recently developed causal inference methods. In addition, we apply it to a real-world dataset on the dynamics of firm growth. We hope, by releasing the full implementation of the method, to facilitate the further development and adoption of this procedure by researchers in a variety of fields.

2 'FCI' in a nutshell

2.1 Problem definition

A directed graph $\mathcal{G}$ is a pair $(\boldsymbol{V}, \boldsymbol{E})$ where $\boldsymbol{V} = \{1, \ldots, M\}$ is a finite set of vertices and $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ is the set of directed edges between the vertices. If $(i, j) \in \boldsymbol{E}$ then we write $i \to j$ and say that i and j are *adjacent*, j is a *child* of i, i is a *parent* of j, and write $i \in \pi(j)$, the parent set of j. A *directed path* from i to j is a sequence of one or more edges $i \to \ldots \to j$ where all edges along the path point towards j. In a *directed acyclic graph* (DAG) there are no directed paths from any vertex to itself. If there exists a directed path from vertex i to j, or if $i = j$, we say that i is an *ancestor* of j, and j is a *descendant* of i.

DAG structures are often used to model data-generating processes, whereby each vertex i of $\mathcal{G}$ represents one random variable X_i. Furthermore,

Pp. 121–129 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

we associate with each vertex i a conditional distribution $P(X_i \mid X_{\pi(i)})$ representing the mechanism by which X_i is generated conditional on the values of its parents in the graph. Such a model is causal if, when a given variable X_i is intervened on and *set* to a specific value, the post-interventional distribution is represented by the same model but with all edges *into* node i removed (Pearl, 2000).

If the data is generated as described above, with DAG $\mathcal{G}$, the joint distribution $P(\boldsymbol{X})$ (where $\boldsymbol{X} = (X_1, \ldots, X_M)$) contains independencies related to the structure of $\mathcal{G}$. This is embodied in the graphical criterion of *d-separation* (Pearl, 1988); if vertices i and j are d-separated given some subset $K \subseteq \boldsymbol{V} \setminus \{i, j\}$ then in the distribution $P(\boldsymbol{X})$ we necessarily have that X_i is independent of X_j given the variables $\{X_k : k \in K\}$, which we write $X_i \perp\!\!\!\perp X_j \mid X_K$. This is known as the *Markov* condition. If, furthermore, *all* independencies in $P(\boldsymbol{X})$ are entailed by d-separation in $\mathcal{G}$, then we say that the distribution $P(X)$ is *faithful* to the graph $\mathcal{G}$.

If the data-generating process results in a distribution faithful to the underlying $\mathcal{G}$, constraint-based search algorithms can be used to infer various aspects of $\mathcal{G}$. In general, causally different graphs can result in the same set of independencies in the data, a phenomenon known as *Markov equivalence*. However, in many cases the set of all graphs which are consistent with the observed pattern of independencies in the data have some features in common, and such features can then be inferred. For *causally sufficient* systems, where all the X_i, $i = 1, \ldots, M$, have been observed, the Markov equivalence classes are particularly simple (Spirtes et al., 2000).

2.2 MAGs, PAGs, and FCI

For the more involved task of finding equivalence classes in causally insufficient systems Spirtes et al. (2000) developed the FCI algorithm. Its output graph contains partial information about *ancestral* relationships among the observed variables and is thus termed a *partial ancestral graph* (PAG).

Towards this end, we need the following definitions (Richardson and Spirtes, 2002). A *mixed graph* is a graph that contains three types of edges: undirected (—), directed (→) and bi-directed (↔). (As we exclude selection bias we will not be dealing with undirected edges so all our subsequent definitions are conditional on this.) There can be at most one such edge between any given pair of vertices, and no edge can connect a vertex to itself. The terms parent, child, directed path, ancestor and descendant are defined as for DAGs. Additionally, if $i \leftrightarrow j$ then i is a *spouse* of j. An *ancestral graph* is a mixed graph for which there is no vertex i which is an ancestor of any of its parents nor any of its spouses.

Ancestral graphs can represent systems derived from DAGs but in which a subset of the variables have been hidden. A graphical separation property termed *m-separation* can be defined, mirroring d-separation for DAGs, such that m-separation in an ancestral graph corresponds to independencies between the observed variables in the distribution. A *maximal ancestral graph* (MAG) is an ancestral graph such that for every pair of variables $\{X_i, X_j\}$ there is an edge between i and j if and only if there does not exist a set $K \subseteq \boldsymbol{V} \setminus \{i, j\}$ such that $X_i \perp\!\!\!\perp X_j \mid X_K$. See (Richardson and Spirtes, 2002).

If two MAGs entail the same set of conditional independencies they are said to be Markov equivalent. Thus, a PAG describes an equivalence class of MAGs, and is a graph with edges having three kinds of edge marks: arrowtails, arrowheads, and circles, in any combination. A PAG $\mathcal{P}$ of an equivalence class fulfills that $\mathcal{P}$ has the same adjacencies as any member of the equivalence class and every non-circle mark is present in all members of the equivalence class. As we exclude selection bias (and hence undirected edges), in our case a PAG can contain the following types of edges: →, ↔, ○→, and ○–○. An edge $X_i \rightarrow X_j$ is interpreted as X_i being an ancestor of X_j (in the underlying DAG), and X_j not being an ancestor of X_i. If $X_i \leftrightarrow X_j$ then neither variable is an ancestor of the other. The circle mark represents cases where, in different members of the equivalence class, both arrowtails and arrowheads are present at this end; hence, based on independencies alone, it is undecided whether that variable is an ancestor or non-ancestor of the other variable.

Finally, the FCI algorithm (Spirtes et al., 2000) uses independence tests on the observed data to infer the appropriate PAG and thus (partial) information on ancestral relationships between the observed variables. We describe the relevant details of the algorithm, along with the necessary adaptations to the time series case in Section 4. For the moment, it suf-

fices to say that the algorithm starts from the complete graph over all variables, and performs a series of independence tests according to which it removes edges between those pairs of variables which are (conditionally) independent. Subsequent 'orientation rules' are employed to derive causal conclusions which, under the stated assumptions and in the limit of correctly identified dependencies and independencies, are guaranteed to be valid. The full algorithm, including a proof that the method not only is sound but that it also is complete, is described in (Zhang, 2008).

3 Time series model

Let $\boldsymbol{X}(t) = (X_1(t), \ldots, X_M(t))$ be a multivariate time series with M variables defined at discrete time points $t \in \mathbb{Z}$, with $\boldsymbol{X}(t)$ either continuous ($\in \mathbb{R}^M$) or discrete-valued ($\in \mathbb{Z}^M$). We assume that the time series data is generated by the following process (essentially a 'dynamic bayes network' with hidden variables, or more precisely a discrete-time first-order time-invariant multivariate Markov process with sparse connections and hidden variables):

1. The causal structure of the time series is defined by a bipartite graph $\mathcal{G}$ with directed edges $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ where $\boldsymbol{V} = \{1, \ldots, M\}$ is the variable set. An edge $(i, j) \in \boldsymbol{E}$ if and only if variable i has a direct causal effect on variable j.

2. For all j and t, the value of $X_j(t)$ is drawn from a distribution $P(X_j(t) \mid \boldsymbol{X}_{\pi(j)}(t-1)) > 0$ where $\pi(j)$ is the set of parents of variable j.[1]

3. Assume that the process has a strictly positive invariant distribution (or density). For discrete variables with a finite state space, this is guaranteed by the positivity of the conditional distribution, whereas for linear-Gaussian systems the absolute values of all eigenvalues of the coefficient matrix need to be smaller than unity.

4. The observed data is a subset of size $N \leq M$ of the generating variables. Without loss of generality, let these be the first N variables, i.e. $X_1(t), \ldots, X_N(t)$. The data is observed for time indices $t = \{1, \ldots, T\}$, and the process is assumed already to be at equilibrium at time $t = 1$ (i.e. we are sampling from equilibrium).

[1] Note that for continuous variables, the conditional distributions are typically represented by conditional *densities*.

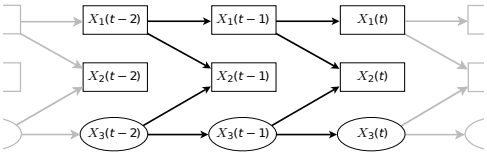


Figure 1: Data generating process over variables $\boldsymbol{X}(t) = (X_1(t), X_2(t), X_3(t))$. The squared nodes represent observed variables, the oval ones hidden.

An example model is shown in Figure 1. In the linear-Gaussian case this model would be represented by $\boldsymbol{X}(t) = \boldsymbol{A}\boldsymbol{X}(t-1) + \varepsilon(t)$, where $\boldsymbol{A}$ is the coefficient matrix and $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ represents Gaussian noise. Note that $\boldsymbol{A}$ only has nonzero entries where the corresponding variables are connected by an edge. In the discrete case the parameters would be conditional probabilities; for binary variables there is one parameter for each variable $X_i(t)$ and each (joint) state of the parents of that variable $X_{\pi(i)}(t-1)$.

The model is quite general: For example, higher-order Markov processes can be represented by transforming them to first-order with additional hidden variables, and in terms of linear systems the model can represent any finite-order ARMA model.

Note that in this model the equilibrium distribution of the time series data over any finite-length time window of length τ, i.e. $P(\boldsymbol{X}(t-\tau), \ldots, \boldsymbol{X}(t))$ is well-defined, and we can obtain (correlated) samples from it by taking windowed samples of the time series data. Furthermore, the nodes corresponding to variable-timepoint pairs satisfy the Markov assumption with regard to the complete unrolled graph (although note that dependencies due to nodes 'outside the window' need to be taken into account). Finally, to be able to utilize the constraint-based causal inference framework, we need to require that the distribution $P(\boldsymbol{X}(t-\tau), \ldots, \boldsymbol{X}(t))$ is faithful to the unrolled graph; that is, it cannot contain independencies that are not represented in the structure of the unrolled graph.

4 FCI for time series

Given a single long sequence $\boldsymbol{X}(t)$, $t = 1, \ldots, T$ of multivariate time series data from a causal time

series model with potentially hidden variables, as defined in Section 3, we can obtain correct (but typically only partial) knowledge about causal relationships in the large sample limit as follows:

First, using the 'sliding window' approach, we transform the original time series data into a set of samples of the random vector $\boldsymbol{X}$ which collects the values of all observed variables within a time-window of finite length τ, i.e. $\boldsymbol{X} = (X_1(t-\tau), \ldots, X_N(t-\tau), \ldots, X_1(t), \ldots, X_N(t))^T$. Note that this random vector is of length $(\tau+1)N$, and we obtain a total of $T-\tau$ samples of it from the data. Since by assumption the observed data come from the time series at equilibrium, each sample is drawn from the equilibrium distribution $P(\boldsymbol{X})$. (Of course, samples coming from close-by timepoints will be correlated, as in any Markov chain, but as $T \to \infty$ the number of effectively independent samples grows without bound.)

Next, considering each component of $\boldsymbol{X}$ as a separate random variable, the FCI algorithm, designed for non-temporal data, is directly applicable. Given successful independence tests (achievable in the large sample limit) we obtain partial but correct causal inferences concerning the elements of the random vector $\boldsymbol{X}$, and hence concerning the original time series.

However, the amount of information returned by standard FCI, even in the large sample limit, can be quite restricted (for an example see Section 5). Fortunately, because we know the data came from a time series process, we have much prior information that we can leverage. In particular, we know that (a) causal effects must go forward in time, and (b) because of time-invariance, if $X_i(t-t_1)$ is an ancestor of $X_j(t-t_2)$, then $X_i(t-t_3)$ must also be an ancestor of $X_j(t-t_4)$ whenever $t_1 - t_2 = t_3 - t_4$. By adapting FCI to explicitly incorporate such background knowledge, we not only are able to make more inferences about the existence or non-existence of causal connections, but those inferences we do make are also more often correct, as the prior knowledge regularizes the problem.

Towards this end, we define an edge between $X_i(t-t_1)$ and $X_j(t-t_2)$ to be *homologous* to an edge between $X_k(t-t_3)$ and $X_l(t-t_4)$ if $i=k$, $j=l$, and $t_1 - t_2 = t_3 - t_4$. Because of the time-invariant structure of the data-generating model we obtain the following two Lemmas.

Lemma 1. *On a finite window of length τ including the variables $\boldsymbol{X}(t-\tau), \ldots, \boldsymbol{X}(t)$, if in $P(\boldsymbol{X})$ we have $X_i(t-t_1) \perp\!\!\!\perp X_j(t-t_2) \mid \{X_k(t-t_3), \ldots, X_l(t-t_K)\}$ with $0 \le t_i \le \tau$, then we also have $X_i(t-t_1+t') \perp\!\!\!\perp X_j(t-t_2+t') \mid \{X_k(t-t_3+t'), \ldots, X_l(t-t_K+t')\}$ for all t' such that $(\max(t_i)-\tau) \le t' \le \min t_i$.*

Proof. Since the distribution is time invariant, on an infinitely long window the claim is true. When only including τ lags, the independencies are guaranteed to hold if all involved nodes (in particular including those in the conditioning set) lie inside the window. This is ensured by the restrictions on t'. □

Lemma 2. *Taking into account the time-invariant underlying structure, in the PAG corresponding to $\boldsymbol{X} = (X_1(t-\tau), \ldots, X_N(t))$ any two homologous edges must contain the same edge marks.*

Proof. This follows directly from the underlying time-invariant structure, because $X_i(t-t_1)$ is an ancestor of $X_j(t-t_2)$ if and only if $X_i(t-t_3)$ is an ancestor of $X_j(t-t_4)$ when $t_1 - t_2 = t_3 - t_4$. □

The *tsFCI* (for 'time series FCI') algorithm is thus obtained by adapting the FCI algorithm using the prior knowledge, as shown in Algorithm 1. The changes to FCI, highlighted in dark-blue italic, follow from Lemmas 1 and 2. In addition, as suggested in Spirtes et al. (2000), we can in all independence tests restrict ourselves to conditioning sets containing no nodes from the present or the future and orient all arrows forward in time. We emphasize that this prior knowledge needs to be applied at all stages of the algorithm; simply post-processing the output of standard FCI would be suboptimal.

Note that Lemma 1 implies that if a (conditional) independence is found between two nodes $X_i(t-t_1)$ and $X_j(t-t_2)$, this not only allows us to remove that edge from the PAG, but it additionally implies that also some homologous edges can be removed (including all such *later* edges). The directionality is crucial here: earlier edges cannot necessarily be removed, because at the early end of the window we may not be able to condition on the necessary conditioning set. This means that in the PAG, there may be *extra edges* showing up in the early part of the time window. If the time window is short these will allow us to make *fewer* inferences but, in the limit, will not cause us to make *incorrect* inferences.

Algorithm 1 tsFCI (sketch).

Note: Adaptation of FCI algorithm, see (Spirtes et al., 2000; Zhang, 2008) for details of the original algorithm.

Input: A multivariate time series dataset with N variables, an integer τ defining the window length.

1. Adjacency Phase

(a) Get the fully connected graph G over all $(\tau+1)N$ nodes.

(b) Let Adj_X be the adjacent nodes of a node X in the graph G. Note that G will be modified in the following loop.

Repeat for $n = 0, 1, 2, \ldots$ (size of conditioning set)

Repeat for every ordered pair (X, Y) of adjacent variables with $\text{Adj}_{X,past} := |\text{Adj}_X - \{Y\} - \{Z : Z \text{ occurs after or in same time slice as } X\}| \geq n$

Repeat for every set $\boldsymbol{Z} \subseteq \text{Adj}_{X,past}$ with n elements:

If $X \perp\!\!\!\perp Y \mid \boldsymbol{Z}$:

Remove the edge between X and Y in G, define $\text{SepSet}_{X,Y} = \boldsymbol{Z}$

Remove every homologous edge between $\tilde{X}$ and $\tilde{Y}$ if the corresponding conditioning set $\tilde{\boldsymbol{Z}}$ is fully contained in the graph (by Lemma 1), and define $\text{SepSet}_{\tilde{X},\tilde{Y}} = \tilde{\boldsymbol{Z}}$

Continue with the next pair.

(c) Try to remove more edges as in FCI by an additional procedure *and use Lemma 1 as in the previous step*.

2. Orientation Phase

(a) *For every edge orient the endpoint in the later time as an arrowhead and for every instantaneous edge orient both ends as arrowheads, by background knowledge.*

(b) Use the complete orientation rule set from Zhang (2008) to orient edge endpoints. *Whenever an edge endpoint is oriented, also orient the same endpoint for every homologous edge (by Lemma 2).* (Note that in this step the "SepSet" from 1(b) are needed.)

Output: A PAG over $(\tau+1)N$ nodes with repeating structure.

5 Simulations

We provide a complete implementation of our algorithm, including code for all of our experiments, at: `http://cs.helsinki.fi/u/entner/tsfci/`

First, to illustrate the theoretical potential (infinite sample size limit) of tsFCI as compared to both Granger causality and standard FCI, consider the example introduced in Figure 1 with window length $\tau = 2$. A Granger-causality approach would correctly infer that the only cause of $X_1(t)$ is $X_1(t-1)$, as this is necessary and sufficient for optimal prediction. However, it would also suggest that there are direct causal connections from *both* variables and *all* lags to $X_2(t)$, due to the hidden variable $\boldsymbol{X}_3$. FCI and tsFCI yield the PAGs of Figure 2(a) and (b), respectively. Both find that $X_2(t-1)$ and $X_2(t-2)$ are not ancestors of (i.e. do not have a causal effect on) $X_2(t)$, and $X_2(t-1)$ is not an ancestor of $X_1(t)$. From tsFCI (only), we additionally see that $X_1(t-1)$ *is* a cause of $X_1(t)$ and of $X_2(t)$ (FCI could here not rule out a hidden common cause as an explanation for the correlation). One might think that a simple 'post-processing' of the output of FCI would be sufficient but, in practice with finite sample data, this is not the case as the output of FCI may not respect the regular time series structure (see Figure 2(c)).

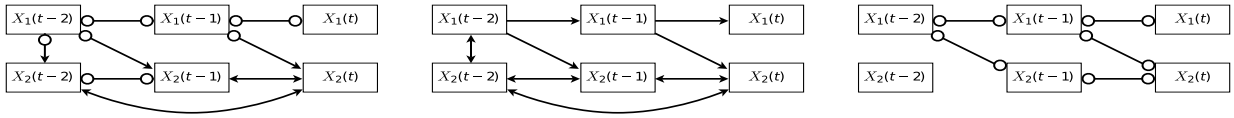
For a comprehensive analysis, we generate data from randomly constructed time series models, and apply FCI, tsFCI, Granger, Group Lasso (GL) for time series (Haufe et al., 2010), and the 'Phase Slope Index' (PSI) method of Nolte et al. (2008) to the data. Both binary and linear-Gaussian data was used; however we did not have implementations of Granger or GL for binary data, and PSI is only applicable to continuous data. We can deduce the behavior of FCI, tsFCI, and Granger in the infinite sample limit by using an independence oracle based on d-separation in the generating model.

The methods are evaluated based on three scores, each measuring the percentages of correct vs incorrect vs "don't know" answers to a set of simple causal queries. The 'direct-cause' score asks if a given node (variable-timepoint pair) has a direct effect on another given node, the 'ancestor' score asks if a given node has any effect (potentially indirect) on another node, and the 'pairwise' score asks if a given variable has an effect on another given variable (with any lag). Note that GL and PSI are specifically constructed to answer the last question.

For reasons of space we cannot give all details of the data generation here, but they are thoroughly described in our online code package. Briefly, we used models with up to 6 observed and 3 hidden time series variables, with edge probabilities $q = 0.25$ and $q = 0.5$ between any pair of nodes, and generated data of three different sample sizes (100, 1,000, 10,000). The window length is fixed at $\tau = 3$ and we set the significance level to 0.01.[2] Results are shown in Figure 3. We state a few main points:

First, in the infinite sample size limit ("$T \to \infty$"),

[2] Simulations showed that the significance level seems not to have a crucial effect on the results. For smaller window lengths τ less decisions are made than for longer ones.



(a) original FCI with oracle (b) tsFCI with oracle (c) original FCI on data

Figure 2: Output PAG of FCI and tsFCI when applying the algorithms to the generative model in Figure 1, in (a) and (b) with an oracle (infinite sample size limit), and in (c) for a small sample size.

the main difference between the FCI-based methods and Granger causality is that with the former some decisions are not made but all made decisions are correct, whereas in the latter all decisions are made but some of them are wrong (because of the latents). In this sense FCI/tsFCI is (in theory) more conservative than Granger. As expected, tsFCI makes somewhat more predictions than FCI.

Second, tsFCI consistently outperforms FCI on finite-length data as well, particularly for continuous variables and the ancestor score (structural mistakes of FCI may be reflected in this case).

Third, on all scores and for all methods, the accuracy increases with sample size, as expected. However, for realistic sample sizes, FCI and tsFCI are quite far from the infinite sample limit. In particular, we found that as the unrolled graphs were relatively dense (even for $q = 0.25$), these methods need to rely on independence tests with large conditioning sets, leading to inevitable errors that accumulate. Only for the sparse, binary case does the performance approach the theoretical limit. For the continuous data, with our test settings, the standard Granger procedure is superior to both FCI-based methods both in terms of number of decisions made *and* in terms of the correctness of those decisions.

Fourth, for binary data, the number of decisions *decrease* for larger samples. This counterintuitive result is due to weak causal effects (over long paths) detected only for these sample sizes, which yields to more edges and hence fewer orientation rules might be applied. This is not particular to our case, but can occur in other applications of FCI (to DAG inference) as well.

Finally, in our experiments neither PSI nor Group Lasso outperformed a basic Granger analysis. Since we tried to choose the parameters close to optimal for these methods, the explanation for our results is that the particularities of the data generating setup favored Granger. For instance, PSI was at a clear disadvantage here because it seeks driver-receiver-relationships among the variables, and so does not allow both variables in an interacting pair being drivers of the other. Such relationships were however common in our data.

6 Firm growth data

We also applied the tsFCI algorithm to microeconomic data of growth rates of US manufacturing firms in terms of employment, sales, research & development (R&D) expenditure, and operating income, for the years 1973–2004. Here, we estimated the model including instantaneous effects,[3] using a significance level of 0.001 and $\tau = 3$. The first two lags of the PAG are shown in Figure 4.

The graph suggests that there are direct causal effects over time from employment to sales growth and from sales to employment growth. Furthermore, there is no causal effect from operating income to R&D growth (instantaneously or lagged), whereas both employment and sales growth have (direct or indirect) lagged effects on R&D growth. Interestingly, these features essentially duplicate the main findings of Moneta et al. (2010), in which a Structural Vector Autoregression model was used to infer causal relationships among the variables. One difference, however, is that their analysis suggested significant contemporaneous causal effects among the variables, while our present approach attributes most of the correlations to latents.

7 Related work

Most standard approaches to causal inference from non-experimental time series data are rooted in the concept of Granger causality (Granger, 1969). In

[3] In Algorithm 1 this means to not exclude nodes from the present in the conditioning sets in step 1(b) and to not orient instantaneous edges as double-headed arrows in step 2(a).

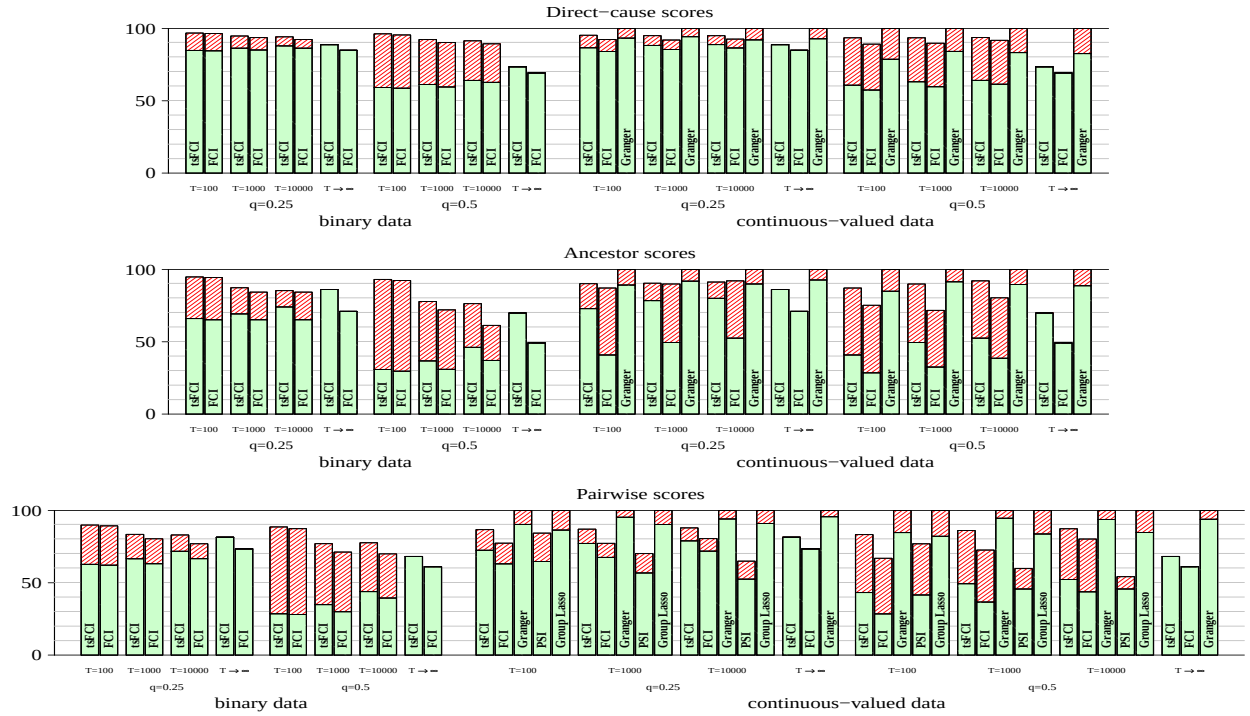


Figure 3: Results of simulations. From top to bottom row: direct-cause score, ancestor score, pairwise score. In each row, the two left blocks of bars show the result for binary data for sparse (q=0.25) and less sparse (q=0.5) graphs for different sample sizes and algorithms. The two right blocks of bars represent the same results for continuous-valued data. The total height of each bar shows the percentage of made decisions. Within each bar the green filled area marks the correct decisions, the red striped area the incorrect decisions.

essence, the assumption is that all relevant variables have been measured, and there are no contemporaneous effects, in which case the problem of determining causality turns into a well-defined statistical estimation problem. Naturally, it is important to note that the chosen algorithms and representations (such as whether to use a time-domain or frequency-domain representation) are important in determining the overall performance of the algorithm from a finite-length time series; see, for instance, the work of Nolte et al. (2008) and Haufe et al. (2010).

One departure from the standard assumptions is to include contemporaneous causation. For linear models, this amounts to using Structural Vector Autoregression (SVAR) models rather than the Vector Autoregression (VAR) model. The extra causal information inherent in SVAR models can be inferred using conditional independence tests (Swanson and Granger, 1997; Demiralp and Hoover, 2003) or utilizing non-Gaussianity (Hyvärinen et al., 2008). Nonlinear additive models, with latents which are uncorrelated over time (Chu and Glymour, 2008), constitute an interesting extension of this line of work. However, these models are not guaranteed in the limit to give correct results when temporally dependent hidden variables may be present.

Another interesting recent development is given by the *difference-based causal models* of Voortman et al. (2010), specifically designed to model systems that can be well represented by difference equations. For such systems, their framework is likely to be superior to more general models in terms of inferring the causal structure.

Finally, the strongest connection to other recent work is to the theoretical work of Eichler (2009), for which, unfortunately, at the time of writing no detailed algorithm was published to compare our method against. In his approach, the starting point is the estimation of Granger-causal vs Granger-non-causal relationships, and any remaining contemporaneous dependencies. From such a *path diagram*, certain causal vs non-causal inferences can be made

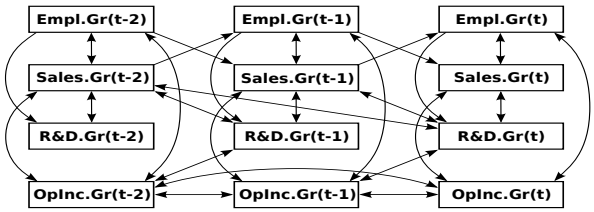


Figure 4: Partial ancestral graph inferred from the firm growth data (see Section 6 for discussion).

which are robust with respect to the presence of hidden variables. His method is also, though much less directly than ours, derived from the theory developed for DAG structures. The main difference between the two approaches is that tsFCI explicitly models the temporal dynamics. This has the advantage of more often being able to detect the absence of direct effects between variable pairs. This comes at a cost, however: For short time series it may be difficult to reliably infer all temporal dependencies, in which case the predictions may be less reliable.

8 Conclusions

While the assumptions underlying FCI seem in many respects more likely to hold than those needed to obtain valid causal conclusions from a Granger-causality approach, our study suggests that caution is in order. Even for processes with relatively few interacting components, the practical problem of detecting which independencies hold (and which do not), from time series of reasonable length, can be a significant obstacle to reliable causal inference. Nevertheless, we suggest that, when the results are interpreted with due caution, tsFCI may help shed light on causal interactions in time series data.

Finally, while the models we consider are very general, there is one respect in which they are quite restricted: We cannot model cyclic contemporaneous causation. In many real world scenarios one may not have a high enough sampling rate to ensure that such causation within a measurement period does not exist, including feedback loops. Unfortunately, such loops are a violation of a basic assumption underlying FCI. Furthermore, many time series are aggregates in the sense that each data point is a sum or average of a number of time slices in an original time series with much faster dynamics. In such cases the independencies present in the measured data do not necessarily reflect the causal structure in the original data. A challenge is how to extend present methods to handle this type of data.

Acknowledgements

Thanks to Alex Coad, Michael Eichler, Stefan Haufe, Alessio Moneta, Guide Nolte, Joseph Ramsey, and Peter Spirtes, for comments, discussions, and making code and data available. Funding for this work was provided by the Academy of Finland.

References

Chu, T. and Glymour, C. (2008). Search for additive nonlinear time series causal models. *JMLR*, 9:967–991.

Demiralp, S. and Hoover, K. D. (2003). Searching for the causal structure of a vector autoregression. *Oxford Bulletin of Economics and Statistics*, 65:745–767.

Eichler, M. (2009). Causal inference from multivariate time series: What can be learned from Granger causality. In *13th International Congress on Logic, Methodology and Philosophy of Science*.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37:424–438.

Haufe, S., Müller, K.-R., Nolte, G., and Krämer, N. (2010). Sparse causal discovery in multivariate time series. *JMLR W&CP*, 6:97–106.

Hyvärinen, A., Shimizu, S., and Hoyer, P. O. (2008). Causal modelling combining instantaneous and lagged effects: an identifiable model based on non-gaussianity. In *Proc. ICML*, pages 424–431.

Moneta, A., Entner, D., Hoyer, P. O., and Coad, A. (2010). Causal inference by independent component analysis with applications to micro- and macroeconomic data. *Jena Economic Research Papers*, pages 2010–2031.

Nolte, G., Ziehe, A., Nikulin, V. V., Schlögl, A., Krämer, N., Brismar, T., and Müller, K.-R. (2008). Robustly estimating the flow direction of information in complex physical systems. *Physical Review Letters*, 100:234101.

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

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

Richardson, T. S. and Spirtes, P. (2002). Ancestral graph markov models. *The Annals of Statistics*, 30(4):962–1030.

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search*. MIT Press, 2nd edition.

Swanson, N. R. and Granger, C. W. J. (1997). Impulse response function based on a causal approach to residual orthogonalization in vector autoregressions. *Journal of the American Statistical Association*, 92:357–367.

Voortman, M., Dash, D., and Druzdzel, M. J. (2010). Learning why things change: the difference-based causality learner. In *Proc. UAI*.

Zhang, J. (2008). On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172:1873–1896.

Variable elimination by factor indexing

Sander Evers, Peter J.F. Lucas
Institute for Computer and Information Sciences
Radboud University Nijmegen
s.evers@cs.ru.nl, peterl@cs.ru.nl

Abstract

It is known that solving an exact inference problem on a Bayesian network with many deterministic nodes can be far cheaper than what would be expected based on its treewidth. In this article, we introduce a novel technique for this, which stores a deterministic node as an array of function values rather than one of probabilities. We propose a variable elimination algorithm, including a new elimination heuristic, that maximally exploits this encoding using *factor indexing*. A preliminary empirical evaluation gives promising results.

1 Introduction

In general, exact inference on a Bayesian network is known to take $O(d^w)$ time, where d is the number of states per variable (assumed that these are the same for each variable) and w is the treewidth of the network's moral graph (Dechter, 1999). In the canonical technique for exact inference, *variable elimination* (Zhang and Poole, 1996), this constraint manifests itself as the minimal size of the largest factor that is created during the execution of the algorithm; implemented as a multidimensional array, it has w dimensions and d entries for each dimension.

When a network contains deterministic nodes, inference can be much faster. One example where this can be seen is the approach of Chavira and Darwiche (2008), in which a Bayesian network is transformed into a logical theory, and inference is performed by counting the models of this theory. These models should be consistent with the constraints imposed by the deterministic nodes. A good model counting algorithm can use these constraints effectively to prune the model search space.

A different approach (Larkin and Dechter, 2003) stays closer to variable elimination. Here, a factor is implemented not as an array (with an entry for each possible variable assignment), but as a list of variable assignments that are nonzero (sometimes called a *sparse array*). The length of this list can be much smaller than the size of the array, but the overhead for multiplying and marginalizing factors is larger, because the list has to be searched for values (possibly using a hash table). With this alternative implementation of factors, ordinary variable elimination can be performed.

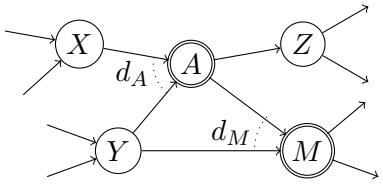


Figure 1: Fragment of a Bayesian network. Variables A and M are deterministic (indicated by a double border): their values follow directly from their parents' values. This is reflected in A's conditional probability distribution: $\mathsf{P}(a|x,y)$ equals 1 if $a = d_A(x,y)$, and 0 otherwise.

The approach we present in this article is also based on a cheaper implementation, but only of the deterministic factors. For example, consider deterministic variable A in Fig. 1. Conventionally, the corresponding factor is stored as a three-dimensional array of conditional probabilities $\mathsf{P}(a|x,y)$, which can take two values: 1 if $a = d_A(x,y)$, 0 if $a \neq d_A(x,y)$. Here, d_A is the function that determines A given X and Y. In our approach, we store this function directly, as a two-dimensional array. If A has n possible states, this array contains n times as few elements as the original array of probabilities.

Pp. 129–137 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

The true merit of our approach, however, lies not in the efficient space use of the arrays that make up the definition of the Bayesian network; it lies in the ability to propagate this efficiency to the intermediate arrays used in variable elimination. For example, consider the elimination of variable A from Fig. 1. In conventional variable elimination, this requires the calculation of $\sum_a \mathsf{P}(a|x,y)\mathsf{P}(z|a)\mathsf{P}(m|a,y)$ for each combination x, y, z, m; a four-dimensional array of which each element is the result of summing n probabilities. On the other hand, our approach exploits the fact that

$$\begin{aligned}&\textstyle\sum_a \mathsf{P}(a|x,y)\mathsf{P}(z|a)\mathsf{P}(m|a,y)\\ &= \mathsf{P}(z|A{=}d_A(x,y))\mathsf{P}(m|A{=}d_A(x,y),y), \quad (1)\end{aligned}$$

i.e. we construct two low-dimensional arrays (over variables XYZ and MXY, resp.) and perform no summation at all. Moreover, we do not even need to multiply these arrays at this point yet; we could perhaps first eliminate Z from the former.

We integrate this operation into variable elimination by expressing it in *factor algebra*, a convenient language to describe the 'bulk operations' on multidimensional arrays that occur in most inference procedures. To the multiplication and summation operations usually encountered, we add an indexing operation written $cpd_Z[A{=}d_A]$ (for the first of the two arrays above). This *partially* indexes the array cpd_Z, which we will define as to contain the probabilities $\mathsf{P}(z|a)$, using *another array*, namely d_A. Its implementation does not require the overhead of sparse arrays, as no additional data structures are used.

The remainder of the article has the following outline. Sect. 2 summarizes the formal preliminaries for inference on Bayesian networks. In Sect. 3, we review variable elimination, with an emphasis on the use of factor algebra. Our main contribution, *factor indexing*, is presented in Sect. 4, followed by an empirical evaluation in Sect. 5. In Sect. 6, we conclude and propose future work.

2 Formal preliminaries

A *Bayesian network* is a triple $(\mathbf{V}, \mathsf{par}, \mathbf{cpd})$. The set $\mathbf{V} = \{V_1, \ldots, V_n\}$ consists of n discrete variables; each V_i has a finite domain $\mathrm{dom}(V_i)$. The function par maps each variable V_j to a set of *parents* $\mathbf{V}_{\mathsf{par}(j)} \subset \mathbf{V}$ in such a way that there are no cycles. The set $\mathbf{cpd} = \{cpd_1, \ldots, cpd_n\}$ contains, for each variable V_j, the family of conditional probability distributions $\mathsf{P}(v_j|\mathbf{v}_{\mathsf{par}(j)})$; in Sect. 3, we will define this family as a *factor* that we designate cpd_j. A Bayesian network defines a joint probability distribution over $\mathbf{V}$: $\mathsf{P}(\mathbf{v}) = \prod_{1\le j\le n} \mathsf{P}(v_j|\mathbf{v}_{\mathsf{par}(j)})$.

An *inference query* is a usually defined as the conditional probability distribution $\mathsf{P}(\mathbf{q}|\mathbf{e})$ over some query variables $\mathbf{Q} \subseteq \mathbf{V}$ given an instantiation $\mathbf{e}$ of evidence variables $\mathbf{E} \subseteq \mathbf{V}$. However, for simplicity we define an inference query as the distribution $\mathsf{P}(\mathbf{q}, \mathbf{e})$ in this article. This distribution can be derived from the joint distribution by summing out the remaining variables $\mathbf{R} = \mathbf{V} \setminus (\mathbf{Q} \cup \mathbf{E})$:

$$\mathsf{P}(\mathbf{q},\mathbf{e}) = \sum_{\mathbf{r}\in\mathbf{R}} \mathsf{P}(\mathbf{q},\mathbf{e},\mathbf{r}) = \sum_{\mathbf{r}\in\mathbf{R}} \prod_{1\le j\le n} \mathsf{P}(v_j|\mathbf{v}_{\mathsf{par}(j)})$$

From this, the conditional distribution can easily be derived: $\mathsf{P}(\mathbf{q}|\mathbf{e}) = \mathsf{P}(\mathbf{q},\mathbf{e}) / \sum_{\mathbf{q}\in\mathbf{Q}} \mathsf{P}(\mathbf{q},\mathbf{e})$.

3 Factor algebra for variable elimination

In this section, we review the theory of factor algebra and variable elimination (Zhang and Poole, 1996), into which we will integrate our new factor indexing operation in Sect. 4. Many inference algorithms, including variable elimination and junction tree propagation (Lauritzen and Spiegelhalter, 1988), are implemented using multidimensional arrays; a *factor* is a mathematical description of such an array, and *factor algebra* is a convenient language to describe array operations.

Formally, a *factor* f over variables $\mathbf{V}$ is a function that maps every instantiation $\mathbf{v}$ of $\mathbf{V}$ to a number $f(\mathbf{v})$; an *instantiation* $\mathbf{v} = \{V_1{=}v_1, \ldots, V_n{=}v_n\}$ is a function mapping each V_j to a value $v_j \in \mathrm{dom}(V_j)$. We refer to the set $\dim(f) = \mathbf{V}$ as f's *dimensionality*. Thus,

each cpd_j in a Bayesian network is a factor with $\dim(cpd_j) = V_j \cup \mathbf{V}_{\mathsf{par}(j)}$ and values

$$cpd_j(V_j{=}v_j, \mathbf{v}_{\mathsf{par}(j)}) = \mathsf{P}(v_j | \mathbf{v}_{\mathsf{par}(j)})$$

The *weight* of a factor f is defined as the number of different instantiations it can be applied to, and equals the size of the array needed to store all f's values:

$$\mathsf{weight}(f) \stackrel{\text{def}}{=} \prod_{V_j \in \dim(f)} |\mathrm{dom}(V_j)|$$

It is also possible to apply a factor to an instantiation $\mathbf{e}$ of a subset of its dimensions ($\mathbf{E} \subset \mathbf{V}$): then the result $f(\mathbf{e})$ is not real number, but a factor over $\mathbf{V} \setminus \mathbf{E}$. We also find it convenient to use instantiations containing *more* variables than are in the factor's domain; in this case, the superfluous variables are just ignored. So, for example, $f(\mathbf{v}, V_{n+1}{=}y) = f(\mathbf{v})$.

An inference query can then be defined as the factor $inf_{\mathbf{Q},\mathbf{E}}$:

$$inf_{\mathbf{Q},\mathbf{E}}(\mathbf{q}, \mathbf{e}) \stackrel{\text{def}}{=} \mathsf{P}(\mathbf{q}, \mathbf{e}) = \sum_{\mathbf{r} \in \mathbf{R}} \prod_{1 \le j \le n} cpd_j(\mathbf{q}, \mathbf{e}, \mathbf{r})$$

where we apply each cpd_j to a lot of superfluous variables. In fact, $inf_{\mathbf{Q},\mathbf{E}}$ is a factor over $\mathbf{Q} \cup \mathbf{E}$ and contains results for all instantiations of $\mathbf{E}$; however, one is usually interested in the result for specific evidence $\mathbf{e}$. Then, the inference goal is to calculate the partial instantiation $inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e})$, a factor over $\mathbf{Q}$.

The basic factor algebra we use contains an operator $\odot$ for multiplying two factors, a unit element $\mathbb{1}$ for $\odot$, and a summation operator $\Sigma_{\mathbf{W}}$ that sums out the $\mathbf{W}$ dimensions of a factor:

$$(f \odot g)(\mathbf{v}) \stackrel{\text{def}}{=} f(\mathbf{v}) \cdot g(\mathbf{v})$$
$$\mathbb{1}() \stackrel{\text{def}}{=} 1$$
$$(\Sigma_{\mathbf{W}} f)(\mathbf{u}) \stackrel{\text{def}}{=} \sum_{\mathbf{w} \in \mathbf{W}} f(\mathbf{u}, \mathbf{w})$$

where we define $\dim(f \odot g) = \dim(f) \cup \dim(g)$, $\dim(\mathbb{1}) = \emptyset$, and $\dim(\Sigma_{\mathbf{W}} f) = \dim(f) \setminus \mathbf{W}$. Note that we use $\mathbf{w}{\in}\mathbf{W}$ to let variable $\mathbf{w}$ range over all possible instantiations of $\mathbf{W}$. In Sect. 4, we extend this basic factor algebra.

Algorithm 1: Variable elimination. Initialize the set of factors f_j to the conditional probability distributions. In each iteration, heuristically choose a variable V_i. From the current set of factors, replace all that have V_i in their domain by their product, with V_i summed out. For the definition of the cost heuristic, see the running text.

Input:

- Bayesian network $(\mathbf{V}, \mathsf{par}, \{cpd_1, \ldots, cpd_n\})$
- evidence $\mathbf{e}$ (instantiation of $\mathbf{E} \subset \mathbf{V}$)
- query variables $\mathbf{Q} \subset \mathbf{V}$

Output: $inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e})$ (a factor over $\mathbf{Q}$)

$\mathbf{W} := \mathbf{V} \setminus (\mathbf{Q} \cup \mathbf{E})$
foreach cpd_j **do** $f_j := cpd_j(\mathbf{e})$

while $\mathbf{W}$ is not empty **do**
  choose $V_i \in \mathbf{W}$ for which the cost of *eliminate*(V_i) is smallest
  eliminate(V_i)
  $\mathbf{W} := \mathbf{W} \setminus \{V_i\}$

$inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e}) := \bigodot\{\text{all remaining } f_j\}$

procedure *eliminate*(V_i)
  $p := \mathbb{1}$
  foreach f_j s.t. $V_i \in \dim(f_j)$ **do**
    $p := p \odot f_j$
    delete f_j
  $f_i := \Sigma_{V_i} p$

Variable elimination is now formulated as a procedure (Alg. 1) that stepwise constructs the factor $inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e})$ out of the set of factors $cpd_1, \ldots, cpd_n$, using above operators. In each step, all the factors in the current set that contain a certain variable V_i are joined together using $\odot$, after which Σ_{V_i} is applied to the result.

As for which V_i to choose next, Alg. 1 uses a greedy heuristic: it always takes the one with minimal cost. We define this cost to be the size of the largest array constructed in this step, i.e. $\mathsf{weight}(p)$ for the final value of p in *eliminate*(V_i). This heuristic is known as the minweight heuristic; other heuristics are also possible, but minweight is known in practice to

Table 1: Laws of factor algebra.

$$f \odot \mathbb{1} = f \tag{2}$$
$$f \odot g = g \odot f \tag{3}$$
$$f \odot (g \odot h) = (f \odot g) \odot h = \bigodot\{f, g, h\} \tag{4}$$
$$\Sigma_V \Sigma_W f = \Sigma_W \Sigma_V f = \Sigma_{V,W} f \tag{5}$$
$$\Sigma_V (f \odot g) = \Sigma_V f \odot g \quad \text{if } V \in \dim(f),\ V \notin \dim(g) \tag{6}$$
$$\Sigma_V (f \odot g) = f \odot \Sigma_V g \quad \text{if } V \notin \dim(f),\ V \in \dim(g) \tag{7}$$
$$\big(f \odot g\big)(\mathbf{e}) = f(\mathbf{e}) \odot g(\mathbf{e}) \tag{8}$$
$$\big(\Sigma_V f\big)(\mathbf{e}) = \Sigma_V f(\mathbf{e}) \quad \text{if } \mathbf{e} \text{ does not instantiate } V \tag{9}$$

perform best when variables have different domain sizes (Kjærulff, 1990). Note that the algorithm can calculate the cost of *eliminate*(V_i) before actually executing it, as it is not necessary to construct the array p to determine $\mathsf{weight}(p)$.

As a matter of fact, although Alg. 1 can certainly be read to perform array operations at factor assignments such as $p := p \odot f_j$ and $f_i := \Sigma_{V_i} p$, it does not have to perform any array operations *at all*. Instead, it can perform a symbolic construction of a new factor algebra expression at these points. In that case, the result of the algorithm is not an array, but a large symbolic expression which can be evaluated at a later stage to produce said array. Thus, the inference procedure is divided into a *search* phase and an *evaluation* phase.

Correctness

Factor algebra is not only a tool for expressing variable elimination concisely, but also for analyzing its correctness. Using some general laws of factor algebra (see Table 1), we can prove that the factor constructed by Alg. 1 indeed corresponds to the factor $inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e})$ specified in the text above. Making use of the definitions of $\odot$ and $\Sigma_{\mathbf{R}}$, we rewrite the definition of $inf_{\mathbf{Q},\mathbf{E}}$ into:

$$inf_{\mathbf{Q},\mathbf{E}} = \Sigma_{\mathbf{R}} \bigodot_{1 \le j \le n} cpd_j$$

Using laws (8) and (9), the instantiation of evidence is pushed into the expression:

$$inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e}) = \Sigma_{\mathbf{R}} \bigodot_{1 \le j \le n} cpd_j(\mathbf{e})$$

Now, we will prove that the following invariant holds at the start of each iteration:

$$inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e}) = \Sigma_{\mathbf{W}} \bigodot \{\text{all remaining } f_j\}$$

For the first iteration, this is trivial, as $\mathbf{W}$ was set to $\mathbf{R}$ and each f_j to $cpd_j(\mathbf{e})$. Next, eliminating a variable V_i corresponds to the following rewriting:

$$\begin{aligned} & \Sigma_{\mathbf{W}} \bigodot \{\text{all remaining } f_j\} \\ = \quad & \Sigma_{\mathbf{W}} \left(\bigodot_{V_i \notin \dim(f_j)} f_j \ \odot \bigodot_{V_i \in \dim(f_j)} f_j \right) \\ = \ & \Sigma_{\mathbf{W} \setminus V_i} \left(\bigodot_{V_i \notin \dim(f_j)} f_j \ \odot \ \Sigma_{V_i} \bigodot_{V_i \in \dim(f_j)} f_j \right) \end{aligned}$$

in which the product is restructured using (3,4) into a group that does not contain V_i and one that does; next, distributive law (7) is used to push the summation over V_i into the expression.

The bottom expression corresponds to the invariant for the next iteration, where $\mathbf{W}$ is set to $\mathbf{W} \setminus \{V_i\}$, and the f_j factors with $V_i \in \dim(f_j)$ have been replaced with

$$f_i := \Sigma_{V_i} \bigodot_{V_i \in \dim(f_j)} f_j$$

After the last loop, $\mathbf{W}$ is empty, so

$$inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e}) = \bigodot \{\text{all remaining } f_j\}$$

which is what we wanted to prove.

4 Factor indexing

This section presents the main contribution of this article: *factor indexing*, and its integration in variable elimination. We propose a new factor algebra operator and laws, and extend Alg. 1 into Alg. 2, which uses them to eliminate deterministic variables.

A variable $Y \in \mathbf{V}$ is called *deterministic* if its value is functionally determined by the value of its parents (here $\mathbf{X} \subset \mathbf{V}$). This means that its conditional probability distribution has the following form:

$$cpd_Y(Y{=}y, \mathbf{x}) = \begin{cases} 1 & \text{if } y = d_Y(\mathbf{x}) \\ 0 & \text{if } y \neq d_Y(\mathbf{x}) \end{cases}$$

where d_Y is a factor over $\mathbf{X}$ with values in $\mathrm{dom}(Y)$, which we call Y's *deterministic factor*.

To account for deterministic variables, we extend the definition of a Bayesian network as follows: next to the set **cpd** of conditional probability distributions for non-deterministic variables $V_1, \ldots, V_m$, we include a set $\mathbf{d}$ of factors for deterministic variables $V_{m+1}, \ldots, V_n$. Factor d_j is then a factor over $\mathsf{par}(V_j)$ with values in $\mathrm{dom}(V_j)$. So, unlike a cpd_j factor, $V_j \notin \dim(d_j)$.

Like d_Y and cpd_Y above, every deterministic factor d has a 'probabilistic representation'. Although we want to keep a factor deterministic whenever possible during variable elimination, sometimes it is unavoidable to translate it to its probabilistic representation. For this, we define the factor algebra operator $\mathbb{1}_{V=d}$:

$$\mathbb{1}_{V=d}(V{=}v, \mathbf{u}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } v = d(\mathbf{u}) \\ 0 & \text{if } v \neq d(\mathbf{u}) \end{cases}$$

where $\dim(d) = \mathbf{U}$, and $\dim(\mathbb{1}_{V=d}) = \{V\} \cup \mathbf{U}$.[1]

So, translating a network with deterministic variables into a conventional one can now be defined as $cpd_j = \mathbb{1}_{V_j=d_j}$ for all $m < j \leq n$.

Consider the nodes A and M in Fig. 1 with the following deterministic factors:

$$d_A(X{=}x, Y{=}y) = x + y$$
$$d_M(A{=}a, Y{=}y) = a \cdot y$$

Conventionally, the probabilistic representations of d_A and d_M are used for variable elimination. For example, when eliminating variable A, the following expression would be constructed:

$$\Sigma_A(\mathbb{1}_{A=d_A} \odot \mathbb{1}_{M=d_M} \odot cpd_Z)$$

[1] From these dimensionalities, it immediately follows that $\mathsf{weight}(\mathbb{1}_{V=d}) = |\mathrm{dom}(V)| \cdot \mathsf{weight}(d)$, so an array storing the probabilistic representation has $|\mathrm{dom}(V)|$ as many elements as one storing the deterministic factor.

Let us focus on the values of sub-expression $\mathbb{1}_{A=d_A} \odot \mathbb{1}_{M=d_M}$, a factor over $XYAM$:

$$(\mathbb{1}_{A=d_A} \odot \mathbb{1}_{M=d_M})(X{=}x, Y{=}y, A{=}a, M{=}m) = \begin{cases} 1 & \text{if } a{=}x{+}y \wedge m{=}a{\cdot}y \\ 0 & \text{otherwise} \end{cases}$$

Note that we can rewrite the condition into $a{=}x{+}y \wedge m{=}(x{+}y){\cdot}y$, so m depends on x instead of a. This suggests that we can rewrite the factor into $\mathbb{1}_{A=d_A} \odot \mathbb{1}_{M=d'_M}$, with

$$d'_M(X{=}x, Y{=}y) = (x{+}y){\cdot}y$$

Similarly, cpd_Z can be transformed into

$$cpd'_Z(X{=}x, Y{=}y, Z{=}z) = \mathsf{P}(Z{=}z \,|\, A{=}x{+}y)$$

Neither d'_M nor cpd'_Z contain variable A anymore. Therefore, after rewriting the variable elimination expression using these new factors, the summation over A can be pushed inwards:

$$\begin{aligned} &\Sigma_A(\mathbb{1}_{A=d_A} \odot \mathbb{1}_{M=d_M} \odot cpd_Z) \\ &= \Sigma_A(\mathbb{1}_{A=d_A} \odot \mathbb{1}_{M=d'_M} \odot cpd'_Z) \\ &= \Sigma_A \mathbb{1}_{A=d_A} \odot \mathbb{1}_{M=d'_M} \odot cpd'_Z \end{aligned}$$

Examining the first factor, we see that

$$\begin{aligned} &(\Sigma_A \mathbb{1}_{A=d_A})(X{=}x, Y{=}y) \\ &= \sum_{a \in \mathrm{dom}(A)} (1 \text{ iff } a{=}x{+}y) \quad = 1 \end{aligned}$$

because for each combination x, y there is only one a s.t. $a{=}x{+}y$. Thus, we conclude that

$$\Sigma_A(\mathbb{1}_{A=d_A} \odot \mathbb{1}_{M=d_M} \odot cpd_Z) = \mathbb{1}_{M=d'_M} \odot cpd'_Z$$

where the right hand side represents a much cheaper way to eliminate A than the left hand side. Note that this is the factor algebra equivalent of Eq. 1 (see *Introduction*).

In order to integrate this rewrite rule into variable elimination, we will now formalize the transformations $d_M \Rightarrow d'_M$ and $cpd_Z \Rightarrow cpd'_Z$ in factor algebra.

We do this by introducing a new operation of the form $f[V{=}d]$, where factor f is *indexed* by factor d in dimension V:

$$f[V{=}d](\mathbf{u}) \stackrel{\text{def}}{=} f(\mathbf{u}, V{=}d(\mathbf{u}))$$

where the dimensionality of resulting factor $f[V{=}d]$ is $(\dim(f) \setminus \{V\}) \cup \dim(d)$— so $\mathbf{u}$ is an instantiation over these variables.

With this operation, we can define the above transformations as follows:

$$d'_M = d_M[A{=}d_A]$$
$$cpd'_Z = cpd_Z[A{=}d_A]$$

Note that, contrary to conventional indexing, the dimensionality of $f[V{=}d]$ can be larger than that of f. For example, $\dim(cpd'_Z) = XYZ$, while $\dim(cpd_Z) = AZ$.

To the laws of factor algebra (Table 1), we add the following:

$$f[V{=}d] = \Sigma_V(f \odot \mathbb{1}_{V=d}) \quad (10)$$
$$(f \odot g)[V{=}d] = f[V{=}d] \odot g[V{=}d] \quad (11)$$
$$\mathbb{1}_{W=f[V=d]} = \mathbb{1}_{W=f}[V{=}d] \quad \text{if } V \neq W \quad (12)$$

Using these, we can generalize the rewrite rule above. The elimination of deterministic variable V_i from a product of factors f_j and deterministic factors d_j can be rewritten as follows:

$$\Sigma_{V_i}\left(\mathbb{1}_{V_i=d_i} \odot \bigodot_{V_j \in \dim(f_j)} f_j \odot \bigodot_{V_j \in \dim(d_j)} \mathbb{1}_{V_j=d_j}\right)$$
$$= \bigodot_{V_j \in \dim(f_j)} f_j[V_i{=}d_i] \odot \bigodot_{V_j \in \dim(d_j)} \mathbb{1}_{V_j=d_j[V_i=d_i]} \quad (13)$$

We apply this in a variable elimination algorithm with factor indexing (Alg. 2). It has the same structure as Alg. 1, but is extended as follows:

- For a deterministic variable V_i, we store d_i instead of $\mathbb{1}_{V_i=d_i}$.
- To eliminate a deterministic variable V_i, we use Eq. 13: we index all currently existing f_j and d_j factors over V_i by $V_i{=}d_i$, and delete d_i itself.
- Not all deterministic variables are eliminated like this: during the elimination of a non-deterministic variable V_i, all deterministic factors over V_i have to be expanded to their probabilistic representation. Also, for a deterministic *evidence* variable, its factor is expanded during initialization.

Algorithm 2: Variable elimination with factor indexing. Next to the set of factors f_j, maintain a set of deterministic factors d_j. When eliminating a deterministic variable V_i (for which d_i still exists), do not replace factors by their product but *index* them by d_i. Note: in the absence of deterministic nodes, the algorithm 'degenerates' to Alg. 1.

Input:

- Bayesian network w/ deterministic nodes $(\mathbf{V}, \mathsf{par}, \{cpd_1, \ldots, cpd_m\}, \{d_{m+1}, \ldots, d_n\})$
- evidence $\mathbf{e}$ (instantiation of $\mathbf{E} \subset \mathbf{V}$)
- query variables $\mathbf{Q} \subset \mathbf{V}$

Output: $inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e})$ (a factor over $\mathbf{Q}$)

$\mathbf{W} := \mathbf{V} \setminus (\mathbf{Q} \cup \mathbf{E})$
foreach cpd_j **do** $f_j := cpd_j(\mathbf{e})$
foreach d_j **do**
  if $V_j \in \mathbf{E}$ **then**
    $f_j := \mathbb{1}_{V_j=d_j}(\mathbf{e})$
  else
    $d_j := d_j(\mathbf{e})$

while $\mathbf{W}$ is not empty **do**
  choose $V_i \in \mathbf{W}$ for which the cost of *eliminate*(V_i) is smallest
  eliminate(V_i)
  $\mathbf{W} := \mathbf{W} \setminus \{V_i\}$

$inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e}) := (\bigodot\{\text{all remaining } f_j\}) \odot \bigodot\{\, \mathbb{1}_{V_j=d_j} \mid \text{all remaining } d_j \,\}$
procedure *eliminate*(V_i)
  if d_i exists **then**
    foreach d_j s.t. $V_i \in \dim(d_j)$ **do**
      $d_j := d_j[V_i = d_i]$
    foreach f_j s.t. $V_i \in \dim(f_j)$ **do**
      $f_j := f_j[V_i = d_i]$
    delete d_i
  else
    $p := \mathbb{1}$
    foreach d_j s.t. $V_i \in \dim(d_j)$ **do**
      $p := p \odot \mathbb{1}_{V_j=d_j}$
      delete d_j
    foreach f_j s.t. $V_i \in \dim(f_j)$ **do**
      $p := p \odot f_j$
      delete f_j
    $f_i := \Sigma_{V_i} p$

The used elimination heuristic is still the *cost* of the next elimination step. However, the definition of this cost is also extended. If V_i has *no* deterministic factor d_j associated with it, the cost is still $\mathsf{weight}(p)$. If it *does*, the cost is

$$\sum_{V_i \in \dim(f_j)} \mathsf{weight}(f_j[V_i{=}d_i]) + \sum_{V_i \in \dim(d_j)} \mathsf{weight}(d_j[V_i{=}d_i])$$

Although space does not permit it here, a correctness proof can also be given for Alg. 2. The invariant is:

$$inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e}) = \Sigma_{\mathbf{W}} \left(\left(\bigodot \{ \text{all remaining } f_j \} \right) \odot \bigodot \left\{ \mathbb{1}_{V_j = d_j} \,\middle|\, \text{all remaining } d_j \right\} \right)$$

5 Empirical evaluation

We have implemented the factor algebra described above in Python, using the package NumPy which provides an n-dimensional array and executes array operations using fast C loops (not unlike MATLAB). The $\odot$ operator directly translates to NumPy's array multiplication, which can handle the situation where the operands have different dimensions. Indexing an array with another array is supported in NumPy as well.

We perform inference on 4 networks with deterministic nodes known from the Bayesian network literature (the students network is from the UAI'08 evaluation track). We also investigated 6 generated networks of 100 nodes, with 30 root nodes and 70 nodes with 2 parents (randomly chosen from earlier generated nodes). Each node has randomly generated probabilities; each of the 70 non-root nodes has a chance of being deterministic, in which case we randomly generate a deterministic function. Each variable has the same domain; between networks, we vary the domain size (2 or 4). Also, we vary the fraction of deterministic nodes (30%, 60%, 90% of the non-root nodes).

For each network, we take medians over 10 runs; in each run, we instantiate 10 randomly chosen[2] evidence variables $\mathbf{e}$ and choose one random query variable Q. Then we use algorithms Alg. 1 and Alg. 2 to generate a symbolic expression (a *plan*) for $inf_{\mathbf{Q},\mathbf{E}}(\mathbf{e})$, i.e. we execute them as a *search phase* as discussed in Sect. 3. As it is completely implemented in Python (without regard for speed), we do not time this phase; its performance would severely distort the overall timing results.

We record the cost of the generated plans, i.e. the summed weight of all the intermediate factors. In the second phase, we evaluate the plans and record the (wall clock) duration. The experiments were performed on a machine with a 3GHz Intel Core2Duo processor and 2GB RAM.

Results are shown in Table 2: the factor indexing technique provides speedups ranging up to 16×. Expectations are confirmed that it works best with a high fraction of deterministic nodes and/or larger domain sizes. However, we noticed that the variance in performance between runs can be high: we suspect that the current heuristic can easily guide the algorithm in the wrong way, and will investigate more robust heuristics in the future.

6 Conclusions and future work

We propose a new variable elimination technique for exact inference on Bayesian networks, in which deterministic variables are eliminated not by summation but by a factor indexing operation. We emphasize the role of factor algebra, which enables (a) a concise definition of the algorithm, (b) a straightforward correctness proof, and (c) a model for defining an elimination order heuristic in terms of the *cost* of array operations. Indeed, our updated heuristic has little to do with the network's graph structure anymore; this is in line with common knowledge that treewidth is not so important for highly deterministic networks.

A preliminary empirical evaluation shows that the technique performs decently on real-world networks (small speedups) and good on randomly generated networks (speedups of 1–16). We expect much room for improvement here: first, by developing heuristics that take into account the *actual* cost of performing

[2] However, for students, we took the 9 easiest evidence files from the UAI'08 evaluation.

Table 2: Experimental results. Numbers are median values over 10 random queries.

network	# vars (det.)	plan cost Alg. 1	plan cost Alg. 2	cost impr. Alg. 1/Alg. 2	eval. time (s) Alg. 1	eval. time (s) Alg. 2	speedup Alg. 1/Alg. 2
munin-1	189 (65)	278M	260M	1.00	6.94	7.91	0.935
munin-4	1041 (411)	23.3M	19.2M	1.22	0.481	0.382	1.25
diabetes	413 (24)	13.2M	13.1M	1.00	0.148	0.151	0.994
students	376 (304)	4.32M	14.7K	293	0.205	0.053	4.13
random-2-30	100 (±21)	16.3K	3.85K	2.91	0.0120	0.0106	1.15
random-2-60	100 (±42)	19.6K	2.47K	5.82	0.0121	0.0088	1.35
random-2-90	100 (±63)	14.6K	0.711K	15.0	0.0117	0.0064	1.90
random-4-30	100 (±21)	6.28M	2.38M	9.23	0.122	0.0536	5.38
random-4-60	100 (±42)	2.27M	49.0K	55.1	0.0504	0.0098	5.39
random-4-90	100 (±63)	4.41M	14.7K	257	0.0908	0.0065	16.3

the different array operations instead of the size of the resulting array; second, by exploiting low-level machine knowledge to decrease these actual costs. For example, current CPUs and GPUs often feature vectorized processing modes, which we expect can be exploited for the bulk array operations of probabilistic inference. When used properly, this might outperform inference techniques for determinism that cannot be expressed as array operations, e.g. (Chavira and Darwiche, 2008; Larkin and Dechter, 2003).

Furthermore, we argue that our technique has much potential for combination with other inference algorithms, e.g. with junction tree propagation (Lauritzen and Spiegelhalter, 1988), recursive conditioning (Darwiche, 2001) and factor decomposition techniques (Heckerman and Breese, 1996; Vomlel, 2002; Díez and Galán, 2003).

Acknowledgements

The authors have been supported by the OCTOPUS project under the responsibility of the Embedded Systems Institute. The OCTOPUS project is partially supported by the Netherlands Ministery of Economic Affairs under the Embedded Systems Institute program.

References

Mark Chavira and Adnan Darwiche. 2008. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799.

Adnan Darwiche. 2001. Recursive conditioning. *Artif. Intell.*, 126(1-2):5–41.

Rina Dechter. 1999. Bucket elimination: A unifying framework for reasoning. *Artif. Intell.*, 113(1-2):41–85.

Francisco Javier Díez and Severino F. Galán. 2003. Efficient computation for the Noisy MAX. *Int. J. Intell. Syst.*, 18(2):165–177.

D. Heckerman and J. S. Breese. 1996. Causal independence for probability assessment and inference using Bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 26(6):826–831.

Uffe Kjærulff. 1990. Triangulation of graphs — algorithms giving small total state space. Technical Report R-90-09, Dept. of Mathematics and Computer Science, Aalborg University.

David Larkin and Rina Dechter. 2003. Bayesian inference in the presence of determinism. In C M Bishop and B J Frey, editors, *Proceedings of Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, USA.

S. L. Lauritzen and D. J. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B*, 50(2):157–224.

Jiří Vomlel. 2002. Exploiting functional dependence in Bayesian network inference. In *UAI02*, pages 528–535.

Nevin Lianwen Zhang and David Poole. 1996. Exploiting causal independence in bayesian network inference. *J. Artif. Intell. Res. (JAIR)*, 5:301–328.

Parameter learning in MTE networks using incomplete data

Antonio Fernández
Dept. of Statistics and Applied Mathematics
University of Almería, Spain
afalvarez@ual.es

Helge Langseth
Dept. of Computer and Information Science
The Norwegian University of Science and Technology
helgel@idi.ntnu.no

Thomas Dyhre Nielsen
Dept. of Computer Science
Aalborg University, Denmark
tdn@cs.aau.dk

Antonio Salmerón
Dept. of Statistics and Applied Mathematics
University of Almería, Spain
antonio.salmeron@ual.es

Abstract

Bayesian networks with mixtures of truncated exponentials (MTEs) are gaining popularity as a flexible modelling framework for hybrid domains. MTEs support efficient and exact inference algorithms, but estimating an MTE from data has turned out to be a difficult task. Current methods suffer from a considerable computational burden as well as the inability to handle missing values in the training data. In this paper we describe an EM-based algorithm for learning the maximum likelihood parameters of an MTE network when confronted with incomplete data. In order to overcome the computational difficulties we make certain distributional assumptions about the domain being modeled, thus focusing on a subclass of the general class of MTE networks. Preliminary empirical results indicate that the proposed method offers results that are inline with intuition.

1 Introduction

One of the major challenges when using probabilistic graphical models for modeling hybrid domains (domains containing both discrete and continuous variables), is to find a representation of the joint distribution that support 1) efficient algorithms for exact inference based on local computations and 2) algorithms for learning the representation from data. In this paper we will consider mixtures of truncated exponentials (MTEs) (Moral et al., 2001) as a candidate framework. MTE distributions allow discrete and continuous variables to be treated in a uniform fashion, and it is well known that the Shenoy-Shafer architecture (Shenoy and Shafer, 1990) can be used for exact inference in MTE networks (Moral et al., 2001). Also, the expressive power of MTEs was demonstrated in (Cobb et al., 2006), where the most commonly used marginal distributions were accurately approximated by MTEs.

Algorithms for learning marginal and conditional MTE distributions from complete data have previously been proposed (Rumí et al., 2006; Romero et al., 2006; Langseth et al., 2010; Langseth et al., 2009). When faced with incomplete data, (Fernández et al., 2010b) considered a data augmentation technique for learning (tree augmented) naive MTE networks for regression, but so far no attempt has been made at learning the parameters of a general MTE network.

In this paper we propose an EM-based algorithm (Dempster et al., 1977) for learning MTE networks from incomplete data. The general problem of learning MTE networks (also with complete data) is computationally very hard (Langseth et al., 2009): Firstly, the sufficient statistics of a dataset is the dataset itself, and secondly, there are no known closed-form equations for finding the maximum likelihood (ML) parameters. In order to circumvent these problems, we focus on domains, where the probability distributions mirror standard parametric

Pp. 137–145 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

families for which ML parameter estimators are known to exist. This implies that instead of trying to directly learn ML estimates for the MTE distributions, we may consider the ML estimators for the corresponding parametric families. Hence, we define a generalized EM algorithm that incorporates the following two observations (corresponding to the M-step and the E-step, respectively): *i*) Using the results of (Cobb et al., 2006; Langseth et al., 2010) the domain-assumed parametric distributions can be transformed into MTE distributions. *ii*) Using the MTE representation of the domain we can evaluate the expected sufficient statistics needed for the ML estimators. For ease of presentation we shall in this paper only consider domains with multinomial, Gaussian, and logistic functions, but, in principle, the proposed learning procedure is not limited to these distribution families. Note that for these types of domains exact inference is not possible using the assumed distribution families.

The remainder of the paper is organized as follows. In Section 2 we give a brief introduction to MTE distributions as well as rules for transforming selected parametric distributions to MTEs. In Section 3 we describe the proposed algorithm, and in Section 4 we present some preliminary experimental results. Finally, we conclude in Section 5 and give directions for future research.

2 Preliminaries

2.1 MTE basics

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 variables $\boldsymbol{X}$ is denoted by $\Omega_{\boldsymbol{X}}$. The MTE model is defined by its corresponding potential and density as follows (Moral et al., 2001):

Definition 1. (MTE potential) *Let $\boldsymbol{W}$ be a mixed n-dimensional random vector. Let $\boldsymbol{Z} = (Z_1, \ldots, Z_d)^T$ and $\mathbf{Y} = (Y_1, \ldots, Y_c)^T$ be the discrete and continuous parts of $\boldsymbol{W}$, respectively, with $c + d = n$. We say that a function $f : \Omega_{\boldsymbol{W}} \mapsto \mathbb{R}_0^+$ is a* Mixture of Truncated Exponentials potential *if for each fixed value $\boldsymbol{Z} \in \Omega_{\boldsymbol{Z}}$ of the discrete variables $\boldsymbol{Z}$, the potential over the continuous variables $\mathbf{Y}$ is defined as:*

$$f(\boldsymbol{y}) = a_0 + \sum_{i=1}^{m} a_i \exp\left\{\mathbf{b}_i^T \boldsymbol{y}\right\}, \qquad (1)$$

for all $\boldsymbol{y} \in \Omega_{\mathbf{Y}}$, where $a_i \in \mathbb{R}$ and $\mathbf{b}_i \in \mathbb{R}^c$, $i = 1, \ldots, m$. We also say that f is an MTE potential if there is a partition $D_1, \ldots, D_k$ of $\Omega_{\mathbf{Y}}$ into hypercubes and in each D_ℓ, f is defined as in Eq. 1. An MTE potential is an MTE density *if it integrates to 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. The following is an example of a conditional MTE density.

$$f(y|x) = \begin{cases} 1.26 - 1.15e^{0.006y} \\ \quad \text{if } 0.4 \leq x < 5,\ 0 \leq y < 13\ , \\ 1.18 - 1.16e^{0.0002y} \\ \quad \text{if } 0.4 \leq x < 5,\ 13 \leq y < 43\ , \\ 0.07 - 0.03e^{-0.4y} + 0.0001e^{0.0004y} \\ \quad \text{if } 5 \leq x < 19,\ 0 \leq y < 5\ , \\ -0.99 + 1.03e^{0.001y} \\ \quad \text{if } 5 \leq x < 19,\ 5 \leq y < 43\ . \end{cases}$$

2.2 Translating standard distributions to MTEs

In this section we will consider transformations from selected parametric distributions to MTE distributions.

2.2.1 The Multinomial Distribution

The conversion from a multinomial distribution into an MTE distribution is straightforward, since a multinomial distribution can be seen as a special case of an MTE (Moral et al., 2001).

2.2.2 The Conditional Linear Gaussian Distribution

In (Cobb et al., 2006; Langseth et al., 2010) methods are described for obtaining an MTE

approximation of a (marginal) Gaussian distribution. Common for both approaches is that the split points used in the approximations depend on the mean value of the distribution being modeled. Consider now a variable X with continuous parents $\boldsymbol{Y}$ and assume that X follows a conditional linear Gaussian distribution:[1]

$$X|\boldsymbol{Y} = \boldsymbol{y} \sim \mathcal{N}(\mu = b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{y}, \sigma^2).$$

In the conditional linear Gaussian distribution, the mean value is a weighted linear combination of the continuous parents. This implies that we cannot directly obtain an MTE representation of the distribution by following the procedures of (Cobb et al., 2006; Langseth et al., 2010); each part of an MTE potential has to be defined on a hypercube (see Definition 1), and the split points can therefore not depend on any of the variables in the potential. Instead we define an MTE approximation by splitting $\Omega_{\boldsymbol{Y}}$ into hypercubes $D_1, \ldots, D_k$, and specifying an MTE density for X for each of the hypercubes. For hypercube D_l the mean of the distribution is assumed to be constant, i.e., $\mu^l = b + w_1\mathrm{mid}_1^l + \cdots + w_j\mathrm{mid}_j^l$, where mid_i^l denotes the midpoint of Y_i in D_l (by defining fixed upper and lower bounds on the ranges of the continuous variables, the midpoints are always well-defined). Thus, finding an MTE representation of the conditional linear Gaussian distribution has been reduced to defining a partitioning $D_1, \ldots, D_k$ of $\Omega_{\boldsymbol{Y}}$ and specifying an MTE representation for a (marginal) Gaussian distribution (with mean μ^l and variance σ^2) for each of the hypercubes D_l in the partitioning.

In the current implementation we define the partitioning of $\Omega_{\boldsymbol{Y}}$ based on equal-frequency binning, and we use BIC-score (Schwarz, 1978) to chose the number of bins. To obtain an MTE representation of the (marginal) Gaussian distribution for each partition in $\Omega_{\boldsymbol{Y}}$ we follow the procedure of (Langseth et al., 2010); four MTE candidates for the domain $[-2.5, 2.5]$ are shown in Figure 1 (no split points are being used, except to define the boundary).

[1]For ease of exposition we will disregard any discrete parent variables in the subsequent discussion, since they will only serve to index the parameters of the function.

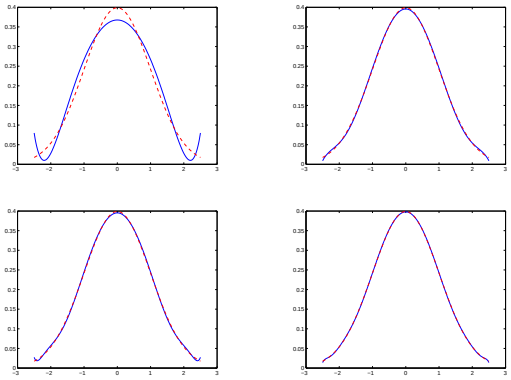


Figure 1: MTE approximations with 5, 7, 9 and 11 exponential terms, respectively, for the truncated standard Gaussian distribution with support $[-2.5, 2.5]$. It is difficult to visually distinguish the MTE and the Gaussian for the three latter models.

Notice that the MTE density is only positive within the interval $[\mu - 2.5\sigma, \mu + 2.5\sigma]$ (confer Figure 1), and it actually integrates up to 0.9876 in that region, which means that there is a probability of 0.0124 of finding points outside this interval. In order to avoid problems with 0 probabilities, we add tails covering the remaining probability mass of 0.0124. More precisely, we define the normalization constant

$$c = \frac{0.0124}{2\left(1 - \int_0^{2.5\sigma} \exp\{-x\}\mathrm{d}x\right)},$$

and include the tail

$$\phi(x) = c \cdot \exp\left\{-(x - \mu)\right\}.$$

for the interval above $x = \mu + 2.5\sigma$ in the MTE specification. Similarly, a tail is also included for the interval below $x = \mu - 2.5\sigma$. The transformation rule from Gaussian to MTE therefore becomes

$$\phi(x) = \begin{cases} c \cdot \exp\left\{x - \mu\right\} & \text{if } x < \mu - 2.5\sigma, \\ \sigma^{-1}\left[a_0 + \sum_{j=1}^{7} a_j \exp\left\{b_j \frac{x-\mu}{\sigma}\right\}\right] & \\ \qquad \text{if } \mu - 2.5\sigma \leq x \leq \mu + 2.5\sigma, & \\ c \cdot \exp\left\{-(x - \mu)\right\} & \text{if } x > \mu - 2.5\sigma. \end{cases} \tag{2}$$

2.2.3 The Logistic Function

The sigmoid function for a discrete variable X with a single continuous parent Y is given by

$$P(X = 1 \mid Y) = \frac{1}{1 + \exp\{b + wy\}}.$$

(Cobb and Shenoy, 2006) propose an 4-piece 1-term MTE representation for this function:

$$P(X = 1 \mid Y = y) = \begin{cases} 0 & \text{if } y < \frac{5-b}{w}, \\ a_0^1 + a_1^1(b,w)\exp\{b^1 w(y - b(w+1))\} & \\ \quad \text{if } \frac{5-b}{w} \leq y \leq \frac{b'}{w}, & \\ a_0^2 + a_1^2(b,w)\exp\{b^2 w(y - b(w+1))\} & \\ \quad \text{if } \frac{b'}{w} < y \leq \frac{-5-b}{w}, 1 \text{ if } y > \frac{-5-b}{w}, & \end{cases} \quad (3)$$

where a_0^k and b^1, b^2 are constants and $a_1^1(b,w)$ and $a_1^2(b,w)$ are derived from b and w. Note that the MTE representation is 0 or 1 if $y < (5-b)/w$ or $y > (-5-b)/w$, respectively. The representation can therefore be inconsistent with the data (i.e., we may have data cases with probability 0), and we therefore replace the 0 and 1 with ϵ and $1-\epsilon$, where ϵ is a small positive number. ($\epsilon = 0.01$ was used in the experiments reported in Section 4.)

In the general case, where X has continuous parents $\boldsymbol{Y} = \{Y_1, \ldots, Y_j\}$ and discrete parents $\boldsymbol{Z} = \{Z_1, \ldots, Z_k\}$, then for each configuration $\boldsymbol{z}$ of $\boldsymbol{Z}$, the conditional distribution of X given $\boldsymbol{Y}$ is given by

$$P(X = 1 \mid \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z} = \boldsymbol{z}) = \frac{1}{1 + \exp\{b_{\boldsymbol{z}} + \sum_{i=1}^{j} w_{i,\boldsymbol{z}} y_i\}}. \quad (4)$$

With more than one continuous variable as argument, the logistic function cannot easily be represented by an MTE having the same structure as in Equation 3. The problem is that the split points would then be (linear) functions of at least one of the continuous variables, which is not consistent with the MTE framework (see Definition 1). Instead we follow the same procedure as for the conditional linear Gaussian distribution: for each of the continuous variables in $\boldsymbol{Y}' = \{Y_2, \ldots Y_j\}$, split the variable Y_i into a finite set of intervals and use the midpoint of the lth interval to represent Y_i in that interval. The intervals for the variables in $\boldsymbol{Y}'$ define a partitioning $D_1, \ldots, D_k$ of $\Omega_{\boldsymbol{Y}'}$ into hypercubes, and for each of these partitions we apply Equation 3. That is, for partition D_l we get

$$P(X = 1 \mid \boldsymbol{y}, \boldsymbol{z}) = \frac{1}{1 + \exp\{b' + w_1 y_1\}},$$

where $b' = b + \sum_{k=2}^{j} \text{mid}_l^k w_l^k$. In the current implementation Y_1 is chosen arbitrarily from $\boldsymbol{Y}$, and the partitioning of the state space of the parent variables is performed as for the conditional linear Gaussian distribution.

3 The General Algorithm

As previously mentioned, deriving an EM algorithm for general MTE networks is computationally hard because the sufficient statistics of the dataset is the dataset itself and there is no closed-form solution for estimating the maximum likelihood parameters. To overcome these computational difficulties we will instead focus on a subclass of MTE networks, where the conditional probability distributions in the network mirror selected distributional families. By considering this subclass of MTE networks we can derive a generalized EM algorithm, where the updating rules can be specified in closed form.

To be more specific, assume that we have an MTE network for a certain domain, where the conditional probability distributions in the domain mirror traditional parametric families with known ML-based updating rules. Based on the MTE network we can calculate the expected sufficient statistics required by these rules (the E-step) and by using the transformations described in Section 2.2 we can in turn update the distributions in the MTE network.

The overall learning algorithm is detailed in Algorithm 1, where the domain in question is represented by the model $\mathcal{B}$. Note that in order to exemplify the procedure we only consider

the multinomial distribution, the Gaussian distribution, and the logistic distribution. The algorithm is, however, easily extended to other distribution classes.

Algorithm 1: An EM algorithm for learning MTE networks from incomplete data.

Input: A parameterized model $\mathcal{B}$ over $X_1, \ldots, X_n$, and an incomplete database $\mathcal{D}$ of cases over $X_1, \ldots, X_n$.

Output: An MTE network $\mathcal{B}'$.

1 Initialize the parameter estimates $\hat{\boldsymbol{\theta}}_{\mathcal{B}}$ randomly.
2 **repeat**
3 Using the current parameter estimates $\hat{\boldsymbol{\theta}}_{\mathcal{B}}$, represent $\mathcal{B}$ as an MTE network $\mathcal{B}'$ (see Section 2.2).
4 **(E-step)** Calculate the expected sufficient statistics required by the M-step using $\mathcal{B}'$.
5 **(M-step)** Use the result of the E-step to calculate new ML parameter estimates $\tilde{\boldsymbol{\theta}}_{\mathcal{B}}$ for $\mathcal{B}$.
6 $\hat{\boldsymbol{\theta}}_{\mathcal{B}} \leftarrow \tilde{\boldsymbol{\theta}}_{\mathcal{B}}$.
7 **until** *convergence* ;
8 **return** $\mathcal{B}'$.

3.1 The EM algorithm

The transformation rules for the conditional linear Gaussian distribution, the multinomial distribution, and the logistic distribution are given in Section 2.2. In order to complete the specification of the algorithm, we therefore only need to define the E-step and the M-step for the three types of distributions being considered.

3.1.1 The M-step

Given a database of cases $\mathcal{D} = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N\}$ we derive the updating rules based on the expected data-complete log-likelihood function Q:

$$\begin{aligned} Q &= \sum_{i=1}^{N} \mathbb{E}[\log f(X_1, \ldots, X_n) \mid \boldsymbol{d}_i] \\ &= \sum_{i=1}^{N} \sum_{j=1}^{n} \mathbb{E}[\log f(X_j \mid \mathrm{pa}(X_j)) \mid \boldsymbol{d}_i] \ . \end{aligned}$$

The updating rules for the parameters for the multinomial distribution and the Gaussian distribution are well-known and can be found in Appendix A (see (Fernández et al., 2010a) for a derivation).

A closed form solution does not exist for the weight vector for the logistic function, and instead one typically resorts to numerical optimization such as gradient ascent for maximizing Q. To ease notation, we shall consider the variable X_j with discrete parents $\boldsymbol{Z}_j$ and continuous parents $\boldsymbol{Y}_j$ (we drop indexes for the parents whenever those are clear from the context). Also, we use $\bar{\boldsymbol{w}}_{\boldsymbol{z},j} = [\boldsymbol{w}_{\boldsymbol{z},j}^{\mathrm{T}}, b_{\boldsymbol{z},j}]^{\mathrm{T}}$ and $\bar{\boldsymbol{y}} = [\boldsymbol{y}^{\mathrm{T}}, 1]^{\mathrm{T}}$, in which case the gradient ascent updating rule can be expressed as

$$\hat{\bar{\boldsymbol{w}}}_{\boldsymbol{z},j} := \bar{\boldsymbol{w}}_{\boldsymbol{z},j} + \gamma \frac{\partial Q}{\partial \bar{\boldsymbol{w}}_{\boldsymbol{z},j}} \ ,$$

where $\gamma > 0$ is a small number and

$$\begin{aligned} \frac{\partial Q}{\partial \bar{\boldsymbol{w}}_{\boldsymbol{z},j}} = \sum_{i=1}^{N} P(\boldsymbol{z} \mid \boldsymbol{d}_i) \Bigg[\int_{\boldsymbol{y}} P(x_j = 1, \bar{\boldsymbol{y}} \mid \boldsymbol{d}_i, \boldsymbol{z}) \\ g_{\boldsymbol{z},x_j=1}(\bar{\boldsymbol{y}}) \bar{\boldsymbol{y}} d\boldsymbol{y} - \\ \int_{\boldsymbol{y}} P(x_j = 0, \bar{\boldsymbol{y}} \mid \boldsymbol{d}_i, \boldsymbol{z}) g_{\boldsymbol{z},x_j=0}(\bar{\boldsymbol{y}}) \bar{\boldsymbol{y}} d\boldsymbol{y} \Bigg] . \end{aligned}$$

In order to find the partial derivative we need to evaluate two integrals. However, the combination of the MTE potential $P(x_j, \bar{\boldsymbol{y}} \mid \boldsymbol{d}_i, \boldsymbol{z})$ and the logistic function $g_{\boldsymbol{z},x_j}(\bar{\boldsymbol{y}})$ makes these integrals difficult to evaluate. In order to avoid this problem we use the MTE representation of the logistic function specified in Section 2.2.3, which allows the integrals to be calculated in closed form.

3.1.2 The E-step

In order to perform the updating in the M-step we need to calculate the following expectations (see Appendix A):

- $\mathbb{E}(X_j \mid \boldsymbol{d}_i, \boldsymbol{z})$
- $\mathbb{E}(X_j \bar{\boldsymbol{Y}} \mid \boldsymbol{d}_i, \boldsymbol{z})$
- $\mathbb{E}(\bar{\boldsymbol{Y}} \bar{\boldsymbol{Y}}^{\mathrm{T}} \mid \boldsymbol{d}_i, \boldsymbol{z})$
- $\mathbb{E}\left[(X_j - \bar{\boldsymbol{l}}_{\boldsymbol{z},j}^{\mathrm{T}} \bar{\boldsymbol{Y}})^2 \mid \boldsymbol{d}_i, \boldsymbol{z}\right]$

All the expectations can be calculated analytically (see Appendix B). The main point to notice in the calculations is that rather than calculating e.g. $\mathbb{E}(\bar{\boldsymbol{Y}}\bar{\boldsymbol{Y}}^{\mathrm{T}} \mid \boldsymbol{d}_i, \boldsymbol{z})$ directly we instead consider each of the components $\mathbb{E}(Y_j Y_k \mid \boldsymbol{d}_i, \boldsymbol{z})$ in the matrix individually.

4 Experimental results

In order to evaluate the proposed learning method we have generated data from the Crops network (Murphy, 1999). We sampled six complete datasets containing 50, 100, 500, 1000, 5000, and 10000 cases, respectively, and for each of the datasets we generated three other datasets with 5%, 10%, and 15% missing data (the data is missing completely at random (Little and Rubin, 1987)), giving a total of 24 training datasets. The actual data generation was performed using WinBUGS (Lunn et al., 2000).

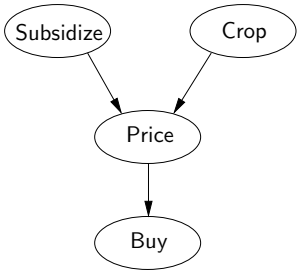


Figure 2: The Crops network.

For comparison, we have also learned baseline models using WinBUGS. However, since WinBUGS does not support learning of multinomial distributions from incomplete data we have removed all cases where Subsidize is missing from the datasets.

The learning results are shown in Table 1, which lists the average (per observation) log-likelihood of the model wrt. a test-dataset consisting of 15000 cases (and defined separately from the training datasets). From the table we see the expected behaviour: As the size of the training data increases, the models tend to get better; as the fraction of the data that is missing increases, the learned models tend to get worse.

The results also show how WinBUGS in general outperforms the algorithm we propose in this paper. We believe that one of the reasons is the way we approximate the tails of the Gaussian distribution in Eq. 2. As the tails are thicker than the actual Gaussian tails, the likelihood is lower in the central parts of the distribution, where most of the samples potentially concentrate. Another possible reason is the way in which we approximate the CLG distribution. Recall that when splitting the domain of the parent variable, we take the average data point in each split to represent the parent, instead of using the actual value. This approximation tends to give an increase in the estimate of the conditional variance, as the approximated distribution needs to cover all the training samples. Obviously, this will later harm the average predictive log likelihood. Two possible solution to this problem are *i*) to increase the number of splits, or *ii*) to use dynamic discretization to determine the optimal way to split the parent's domain. However, both solutions come with a cost in terms of increased computational complexity, and we consider the tradeoff between accuracy and computational cost as an interesting topic for future research.

The algorithm has been implemented in Elvira (Elvira Consortium, 2002) and the software, the datasets used in the experiments, and the WinBUGS specifications are all available from `http://elvira.ual.es/MTE-EM.html`.

5 Conclusion

In this paper we have proposed an EM-based algorithm for learning MTE networks from incomplete data. In order to overcome the computational difficulties of learning MTE distributions, we focus on a subclass of the MTE networks, where the distributions are assumed to mirror known parametric families. This subclass supports a computationally efficient EM algorithm. Preliminary empirical results indicate that the method learns as expected, although not as well as WinBUGS. In particular, our method seems to struggle when the portion of the the data that is missing increases. We have proposed some remedial actions to this problem that we will investigate further.

No. Cases	ELVIRA Percentage of missing data				WINBUGS Percentage of missing data			
	0%	5 %	10%	15%	0%	5 %	10%	15%
50	-3.8112	-3.7723	-3.8982	-3.8553	-3.7800	-3.7982	-3.7431	-3.6861
100	-3.7569	-3.7228	-3.9502	-3.9180	-3.7048	-3.7091	-3.7485	-3.7529
500	-3.6452	-3.6987	-3.7972	-3.8719	-3.6272	-3.6258	-3.6380	-3.6295
1 000	-3.6325	-3.7271	-3.8146	-3.8491	-3.6174	-3.6181	-3.6169	-3.6179
5 000	-3.6240	-3.6414	-3.8056	-3.9254	-3.6136	-3.6141	-3.6132	-3.6144
10 000	-3.6316	-3.6541	-3.7910	-3.8841	-3.6130	-3.6131	-3.6131	-3.6135

Table 1: The average log-likelihood for the learned models, calculated per observation on a separate test set.

Acknowledgments

This work is supported by a grant from Iceland, Liechtenstein, and Norway through the EEA Financial Mechanism. Supported and Coordinated by Universidad Complutense de Madrid. Partially supported by the Spanish Ministry of Science and Innovation, through project TIN2007-67418-C03-02, and by EFDR funds.

A Updating rules

The updating rules for the parameters for the multinomial distribution (i.e., $\theta_{j,k,\boldsymbol{z}} = P(X_j = k|\boldsymbol{Z} = \boldsymbol{z})$) and the conditional linear Gaussian distribution (i.e., $X_j|\boldsymbol{Z} = \boldsymbol{z}, \boldsymbol{Y} = \boldsymbol{y} \sim \mathcal{N}(\bar{\boldsymbol{l}}^{\mathrm{T}}_{\boldsymbol{z},j}\bar{\boldsymbol{y}}, \sigma^2_{\boldsymbol{z},j})$) are given by

$$\hat{\bar{\boldsymbol{l}}}_{\boldsymbol{z},j} \leftarrow \left[\sum_{i=1}^{N} f(\boldsymbol{z} \mid \boldsymbol{d}_i)\mathbb{E}(\bar{\boldsymbol{Y}}\bar{\boldsymbol{Y}}^{\mathrm{T}} \mid \boldsymbol{d}_i, \boldsymbol{z})\right]^{-1} \left[\sum_{i=1}^{N} f(\boldsymbol{z} \mid \boldsymbol{d}_i)\mathbb{E}(X_j\bar{\boldsymbol{Y}} \mid \boldsymbol{d}_i, \boldsymbol{z})\right]$$

$$\hat{\sigma}_{\boldsymbol{z},j} \leftarrow \left[\frac{1}{\sum_{i=1}^{N} f(\boldsymbol{z} \mid \boldsymbol{d}_i)} \sum_{i=1}^{N} f(\boldsymbol{z} \mid \boldsymbol{d}_i)\mathbb{E}\left[(X_j - \bar{\boldsymbol{l}}^{\mathrm{T}}_{\boldsymbol{z},j}\bar{\boldsymbol{Y}})^2 \mid \boldsymbol{d}_i, \boldsymbol{z}\right]\right]^{1/2}$$

$$\hat{\theta}_{j,k,\boldsymbol{z}} \leftarrow \frac{\sum_{i=1}^{N} P(X_j = k, \boldsymbol{Z} = \boldsymbol{z} \mid \boldsymbol{d}_i)}{\sum_{k=1}^{|\mathrm{sp}(X_j)|}\sum_{i=1}^{N} P(X_j = k, \boldsymbol{Z} = \boldsymbol{z} \mid \boldsymbol{d}_i)}$$

B Expected sufficient statistics

To illustrate the calculation of the expected sufficient statistics we consider the calculation of $\mathbb{E}\left[(X_j - \bar{\boldsymbol{l}}^{\mathrm{T}}_{\boldsymbol{z},j}\bar{\boldsymbol{Y}})^2 \mid \boldsymbol{d}_i, \boldsymbol{z}\right]$ (see Section 3.1.2):

$$\mathbb{E}\big[(X_j - \bar{\boldsymbol{l}}^{\mathrm{T}}_{\boldsymbol{z},j}\bar{\boldsymbol{Y}})^2 \mid \boldsymbol{d}_i\big] = \mathbb{E}[X_j^2 \mid \boldsymbol{d}_i] - 2\bar{\boldsymbol{l}}^{\mathrm{T}}_{\boldsymbol{z},j}\mathbb{E}[X_j\bar{\boldsymbol{Y}} \mid \boldsymbol{d}_i] + \mathbb{E}[(\bar{\boldsymbol{l}}^{\mathrm{T}}_{\boldsymbol{z},j}\bar{\boldsymbol{Y}})^2 \mid \boldsymbol{d}_i]$$

For the second component in the summation we need to calculate a vector of expectations, where the kth element is $\mathbb{E}[X_jY_k \mid \boldsymbol{d}_i]$. By letting the ranges of X_j and Y_k be $[x_a, x_b]$ and $[y_a, y_b]$ (dropping the j and k indices for simplicity), respectively, it is easy to show that the expectation can be calculated on closed form:

$$\mathbb{E}\left[X_jY_i \mid \boldsymbol{d}_i\right] = \frac{a_0}{4}(y_b^2 - y_a^2)(x_b^2 - x_a^2) + \sum_{j=1}^{m}\frac{a_j}{{c_j}^2{b_j}^2}\Big(-\exp\{b_jy_a\} + b_jy_a\exp\{b_jy_a\} + \exp\{b_jy_b\} - b_jy_b\exp\{b_jy_b\}\Big)\Big(-\exp\{c_jx_a\} + c_jx_a\exp\{c_jx_a\} + \exp\{c_jx_b\} - c_jx_b\exp\{c_jx_b\}\Big).$$

For $\mathbb{E}[X_j^2 \mid \boldsymbol{d}_i]$ and $\mathbb{E}\left[(\bar{\boldsymbol{l}}^{\mathrm{T}}_{\boldsymbol{z},j}\bar{\boldsymbol{Y}})^2 \mid \boldsymbol{d}_i\right]$ the calculations are similar; for the latter it immediately follows from $\mathbb{E}\left[(\bar{\boldsymbol{l}}^{\mathrm{T}}_{\boldsymbol{z},j}\bar{\boldsymbol{Y}})^2 \mid \boldsymbol{d}_i\right] = \bar{\boldsymbol{l}}^{\mathrm{T}}_{\boldsymbol{z},j}\mathbb{E}\left[\bar{\boldsymbol{Y}}\bar{\boldsymbol{Y}}^{\mathrm{T}} \mid \boldsymbol{d}_i\right]\bar{\boldsymbol{l}}_{\boldsymbol{z},j}$.

References

Barry R. Cobb and Prakash P. Shenoy. 2006. Inference in hybrid Bayesian networks with mixtures

of truncated exponentials. *International Journal of Approximate Reasoning*, 41(3):257–286.

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

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

Elvira Consortium. 2002. Elvira: An environment for creating and using probabilistic graphical models. In José A. Gámez and Antonio Salmerón, editors, *First European Workshop on Probabilistic Graphical Models*, pages 222–230.

Antonio Fernández, Helge Langseth, Thomas Dyhre Nielsen, and Antonio Salmerón. 2010a. MTE-based parameter learning using incomplete data. Technical report, Department of Statistics and Applied Mathematics, University of Almeria, Spain. `http://www.ual.es/~afa109/downloads/Fernandez2010.pdf`.

Antonio Fernández, Jens D. Nielsen, and Antonio Salmerón. 2010b. Learning Bayesian networks for regression from incomplete databases. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 18:69–86.

Helge Langseth, Thomas D. Nielsen, Rafael Rumí, and Antonio Salmerón. 2009. Maximum likelihood learning of conditional MTE distributions. In *Tenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 5590 of *Lecture Notes in Artificial Intelligence*, pages 240–251. Springer-Verlag, Berlin, Germany.

Helge Langseth, Thomas D. Nielsen, Rafael Rumí, and Antonio Salmerón. 2010. Parameter estimation and model selection in mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 51:485–498.

Steffen L. Lauritzen. 1992. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108.

Uri Lerner, Eran Segal, and Daphne Koller. 2001. Exact inference in networks with discrete children of continuous parents. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 319–328, San Francisco, CA. Morgan Kaufmann Publishers.

Roderick J. A. Little and Donald B. Rubin. 1987. *Statistical analysis with missing data*. John Wiley & Sons, New York.

David Lunn, Andrew Thomas, Nicky Best, and David J. Spiegelhalter. 2000. WinBUGS - a Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10(4):325–337.

Serafín Moral, Rafael Rumí, and Antonio Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In *Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 2143 of *Lecture Notes in Artificial Intelligence*, pages 145–167. Springer-Verlag, Berlin, Germany.

Kevin P. Murphy. 1999. A variational approximation for Bayesian networks with discrete and continuous latent variables. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475, San Francisco, CA.

Vanessa Romero, Rafael Rumí, and Antonio Salmerón. 2006. Learning hybrid Bayesian networks using mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 42:54–68.

Rafael Rumí, Antonio Salmerón, and Serafin Moral. 2006. Estimating mixtures of truncated exponentials in hybrid Bayesian networks. *TEST: An Official Journal of the Spanish Society of Statistics and Operations Research*, 15(2):397–421, September.

Gideon Schwarz. 1978. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464.

Prakash P. Shenoy and Glenn Shafer. 1990. Axioms for probability and belief-function propagation. In *Proceedings of the Sixth Workshop on Uncertainty in Artificial Intelligence*, pages 169–198.

Dealing with uncertainty in Gaussian Bayesian networks from a regression perspective

Miguel A. Gómez-Villegas[a], Paloma Main[a], Hilario Navarro[b] and Rosario Susi[a]
[a]Universidad Complutense Madrid, Spain
[b]Universidad Nacional de Educación a Distancia, Spain

Abstract

Some sensitivity analyses have been developed to evaluate the impact of uncertainty about the mean vector and the covariance matrix that specify the joint distribution of the variables in the nodes of a Gaussian Bayesian network (GBN). Nevertheless, uncertainty about the alternative conditional specification of GBN based on the regression coefficients of each variable given its parents in the directed acyclic graph (DAG), has received low attention in the literature. In this line, we focus on evaluating the effect of regression coefficients misspecification by means of the Kullback-Leibler (KL) divergence.

1 Introduction

GBNs are defined as Bayesian networks (BNs) where the joint probability density of $\mathbf{X} = (X_1, X_2, ..., X_p)^T$ is a multivariate normal distribution $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu}$ the $p-$dimensional mean vector and $\boldsymbol{\Sigma}$ the $p \times p$ positive definite covariance matrix using a directed acyclic graph (DAG) to represent the dependence structure of the variables.

As in BNs, the joint density can be factorized using the conditional probability densities of X_i $(i = 1, ..., p)$ given its parents in the DAG, $pa(X_i) \subset \{X_1, ..., X_{i-1}\}$. These are univariate normal distributions with densities

$$f(x_i|pa(x_i)) \sim N(x_i|\mu_i + \sum_{j=1}^{i-1} b_{ji}(x_j - \mu_j), v_i)$$

being μ_i the marginal mean of X_i, b_{ji} the regression coefficients of X_i given $X_j \in pa(X_i)$, and v_i the conditional variance of X_i given its parents in the DAG. Note that if $b_{ji} = 0$ then X_j is not a parent of X_i.

The parameters of the joint distribution can be obtained from the previous conditional specification. More concretely, the means $\{\mu_i\}$ are obviously the elements of the $p-$dimensional mean vector $\boldsymbol{\mu}$ and

$$\boldsymbol{\Sigma} = [(\mathbf{I}_p - \mathbf{B})^{-1}]^T \mathbf{D} (\mathbf{I}_p - \mathbf{B})^{-1}$$

(Shachter and Kenley, 1989) where $\mathbf{D}$ is a diagonal matrix $\mathbf{D} = diag(\mathbf{v})$ with the conditional variances $\mathbf{v}^T = (v_1, ..., v_p)$ and $\mathbf{B}$ a strictly upper triangular matrix with the regression coefficients $b_{ji}, j \in \{1, ..., i-1\}$.

The specification based on $(\boldsymbol{\mu}, \mathbf{B}, \mathbf{D})$ is more manageable for experts because they only have to describe univariate distributions. Moreover, the DAG can be improved by adding the numerical values of the regression coefficient and conditional variance to the corresponding arc and node, respectively.

Nevertheless, there can still be considerably uncertainty about parameters. Sensitivity analysis is an important phase of any modelling procedure. (Castillo and Kjærulff, 2003) performed a one-way sensitivity analysis investigating the impact of small changes in the network parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Alternatively, (Gómez-Villegas, Main and Susi, 2007) proposed a one-way global sensitivity analysis instead of considering local aspects as location and dispersion, over the network's output. Also, in (Gómez-Villegas, Main and Susi, 2008) a $n-$way sensitivity analysis is presented as a generalization of

[1] *E-mail addresses:* ma.gv@mat.ucm.es (M.A. Gómez-Villegas), pmain@mat.ucm.es (P. Main), hnavarro@ccia.uned.es (H. Navarro), rsusi@estad.ucm.es (R. Susi)

Pp. 145–153 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

the previous one both using the KL divergence to evaluate the impact of perturbations.

It is well known the KL divergence is an non-symmetric measure that evaluates the amount of information available to discriminate between two probability distributions. We have chosen it because we want to compare the global behaviors of two probability distributions.

The directed KL divergence between the probability densities $f(w)$ and $f'(w)$, defined over the same domain is

$$D_{KL}(f' \mid f) = \int_{-\infty}^{\infty} f(w) \ln \frac{f(w)}{f'(w)} dw \ .$$

The expression for multivariate normal distributions is given by

$$D_{KL}(f' \mid f) = \\ = \frac{1}{2}\left[\ln \frac{|\boldsymbol{\Sigma}'|}{|\boldsymbol{\Sigma}|} + tr\left(\boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}'\right)^{-1} - \mathbf{I}_p\right)\right] + \\ + \frac{1}{2}\left[\left(\boldsymbol{\mu}' - \boldsymbol{\mu}\right)^T \left(\boldsymbol{\Sigma}'\right)^{-1} \left(\boldsymbol{\mu}' - \boldsymbol{\mu}\right)\right],$$

where f and f' are densities of normal distributions $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $N_p(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ respectively.

In general, if $\mathbf{X} = (X_1, X_2, ..., X_p)^T$ is a random vector normally distributed with parameters $(\mathbf{0}, \boldsymbol{\Sigma})$ where the covariance matrix is inaccurately specified, the effect of a perturbation $\boldsymbol{\Lambda}_{p\times p}$ measured in terms of a directed Kullback-Leibler divergence can be expressed as follows

$$D_{KL}(f|f^{\boldsymbol{\Sigma}}) = \frac{1}{2}[\ln \frac{|\boldsymbol{\Sigma}|}{|\boldsymbol{\Sigma}+\boldsymbol{\Lambda}|} + tr(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Sigma}+\boldsymbol{\Lambda}))]$$

$$= -\frac{1}{2}[\ln|\mathbf{I}_p + \boldsymbol{\Lambda}\boldsymbol{\Sigma}^{-1}| - tr(\boldsymbol{\Lambda}\boldsymbol{\Sigma}^{-1})] \leq$$

$$\leq \frac{1}{4} tr(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Lambda})^2 = \frac{1}{4}||\boldsymbol{\Sigma}^{-1}\boldsymbol{\Lambda}||_F^2 =$$

$$= \frac{1}{4}\sum_{i=1}^{p} \boldsymbol{\lambda}_i^{\mathbf{T}} (\boldsymbol{\Sigma}^{-1})^2 \boldsymbol{\lambda}_i$$

being $f^{\boldsymbol{\Sigma}}$ the density function with the perturbed covariance matrix $\boldsymbol{\Sigma}+\boldsymbol{\Lambda}$, $\boldsymbol{\lambda}_i$ $(i = 1, ..., p)$ each of the columns of $\boldsymbol{\Lambda}$, with the necessary restrictions to get symmetric and positive definite matrices $\boldsymbol{\Sigma}+\boldsymbol{\Lambda}$ and $\boldsymbol{\Lambda}\boldsymbol{\Sigma}^{-1}$, $||\cdot||_F$ the Frobenius matrix norm and $tr(\cdot)$ the trace function.

However, as the directed KL divergence $D_{KL}(f' \mid f)$ can be interpreted as the information lost when f' is used to approximate f, in the following sensitivity analyses f has to be the original model and f' the perturbed one opposite to previously used divergence.

Herein we focus on the repercussion of a misspecified $\mathbf{B}_{p\times p}$ matrix, while the rest of the conditional parameters are known. To evaluate perturbation effects, the proper directed KL divergence is used in all the studied cases.

The paper is organized as follows. In Section 2 the problem is stated and analyzed for constant errors. In Section 3 random perturbations are considered; some examples illustrate the behavior of the proposed measure for both local and global uncertainty.

2 Fixed misspecification of B

Let f be the density of a multivariate normal distribution $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with conditional parameters $\boldsymbol{\mu}, \mathbf{B}$ and $\mathbf{D}$. Denoting by $\boldsymbol{\Delta}_B$ the matrix with additive perturbations on $\mathbf{B}$ and $f^{\mathbf{B}}$ the corresponding perturbed density $N_p\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}^{\boldsymbol{\Delta}_\mathbf{B}}\right)$, where

$$\boldsymbol{\Sigma}^{\boldsymbol{\Delta}_\mathbf{B}} \equiv \left[(\mathbf{I}_p - \mathbf{B} - \boldsymbol{\Delta}_\mathbf{B})^{-1}\right]^T \mathbf{D}(\mathbf{I}_p - \mathbf{B} - \boldsymbol{\Delta}_\mathbf{B})^{-1}$$

(see (Susi, Navarro, Main and Gómez-Villegas, 2009)), the KL divergence for comparing two covariance matrices when the means are equal is

$$D_{KL}^{\mathbf{B}}\left(f^{\mathbf{B}} \mid f\right) = \frac{1}{2}\left[trace\left(\boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}^{\boldsymbol{\Delta}_\mathbf{B}}\right)^{-1}\right) - p\right]. \tag{1}$$

It should be noted that as $\mathbf{I}_p - \mathbf{B}$ and $\mathbf{I}_p - \mathbf{B} - \boldsymbol{\Delta}_\mathbf{B}$ are upper triangular matrices with diagonal entries equal to one then $\ln \frac{|\boldsymbol{\Sigma}^{\boldsymbol{\Delta}_\mathbf{B}}|}{|\boldsymbol{\Sigma}|} = 0$.

Now, given that

$$\boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}^{\boldsymbol{\Delta}_\mathbf{B}}\right)^{-1} = \mathbf{I}_p - \left[(\mathbf{I}_p - \mathbf{B})^{-1}\right]^T \boldsymbol{\Delta}_\mathbf{B}^{\mathbf{T}} + \\ -\boldsymbol{\Sigma}\boldsymbol{\Delta}_\mathbf{B}(\mathbf{I}_p - \mathbf{B})^{-1}\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}\boldsymbol{\Delta}_\mathbf{B}\mathbf{D}^{-1}\boldsymbol{\Delta}_\mathbf{B}^{\mathbf{T}}$$

and

$$trace\left(\left[(\mathbf{I}_p - \mathbf{B})^{-1}\right]^T \boldsymbol{\Delta}_\mathbf{B}^{\mathbf{T}}\right) = \\ = trace\left(\boldsymbol{\Delta}_\mathbf{B}(\mathbf{I}_p - \mathbf{B})^{-1}\right),$$

$$trace\left(\mathbf{\Sigma\Delta_B}(\mathbf{I}_p-\mathbf{B})^{-1}\mathbf{\Sigma}^{-1}\right)=$$
$$=trace\left(\mathbf{\Sigma}^{-1}\mathbf{\Sigma\Delta_B}(\mathbf{I}_p-\mathbf{B})^{-1}\right)$$

the divergence in (1) can be restored as

$$D_{KL}^{\mathbf{B}}\left(f^{\mathbf{B}}\mid f\right)=$$
$$=\frac{1}{2}[p-2trace\left(\mathbf{\Delta_B}(\mathbf{I}_p-\mathbf{B})^{-1}\right)+$$
$$+trace\left(\mathbf{\Sigma\Delta_B}\mathbf{D}^{-1}\mathbf{\Delta_B^T}\right)-p]=$$
$$=\frac{1}{2}\left[tr\left(\mathbf{\Sigma\Delta_B}\mathbf{D}^{-1}\mathbf{\Delta_B^T}\right)\right]=$$
$$=\frac{1}{2}\left[tr\left(\mathbf{\Delta_B^T\Sigma\Delta_B}\mathbf{D}^{-1}\right)\right]. \qquad (2)$$

Note that $(\mathbf{I}_p-\mathbf{B})^{-1}$ is an upper triangular matrix with diagonal entries equal to one and $\mathbf{\Delta_B}(\mathbf{I}_p-\mathbf{B})^{-1}$ is an upper triangular matrix with diagonal entries zero.

Under the same assumptions, if $\boldsymbol{\delta}_{(i)}$ denotes an $(i-1)$-dimensional vector of *local* errors in node i —produced by an erroneous estimation or elicitation of the node i with its parents relationships, the perturbation matrix on $\mathbf{B}$, with only this error source is

$$\mathbf{\Delta_{B,i}}=\begin{pmatrix} & & & \boxed{\mathbf{i}} & & \\ 0 & \cdots & 0 & \boldsymbol{\delta}_{(i)1} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \boldsymbol{\delta}_{(i)i-1} & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

Thus, denoting $f^{\mathbf{B},i}$ as the density with coefficients matrix $\mathbf{B}+\mathbf{\Delta_{B,i}}$, the effect on the joint distribution can be expressed by

$$D_{KL}^{\mathbf{B},i}\left(f^{\mathbf{B},i}\mid f\right)=\frac{1}{2}\left[tr\left(\mathbf{\Delta_{B,i}^T\Sigma\Delta_{B,i}}\mathbf{D}^{-1}\right)\right]=$$
$$=\frac{1}{2v_i}\sum_{k=1}^{i-1}\boldsymbol{\delta}_{(i)k}tr\left(\boldsymbol{\sigma}_{(i-1)k}\left(\boldsymbol{\delta}_{(i)k}\right)^T\right), \qquad (3)$$

$i=2,....,p$, being $\boldsymbol{\sigma}_{(i-1)k}$ the *k-th* column of the submatrix $\mathbf{\Sigma}_{(i-1)}$ built with the first $i-1$ rows and columns of $\mathbf{\Sigma}$. The local perturbation divergence in (3) also may be written in the form:

$$D_{KL}^{\mathbf{B},i}\left(f^{\mathbf{B},i}\mid f\right)=\frac{1}{2v_i}\sum_{k=1}^{i-1}\boldsymbol{\delta}_{(i)k}\left\langle\boldsymbol{\sigma}_{(i-1)k},\boldsymbol{\delta}_{(i)}\right\rangle=$$
$$=\frac{1}{2v_i}\boldsymbol{\delta}_{(i)}^T\mathbf{\Sigma}_{(i-1)}\boldsymbol{\delta}_{(i)}=$$
$$=\frac{1}{2v_i}\left\|\mathbf{U}_{(i-1)}\boldsymbol{\delta}_{(i)}\right\|_2^2$$

with $\mathbf{U}=(\mathbf{I}_p-\mathbf{B})^{-1}$ and the submatrix $\mathbf{U}_{(i-1)}$ determined by the first $i-1$ rows and columns of $\mathbf{U}$.

Returning to the measure of interest, $D_{KL}^{\mathbf{B}}\left(f^{\mathbf{B}}\mid f\right)$ in (2), it can be immediately obtained that

$$D_{KL}^{\mathbf{B}}\left(f^{\mathbf{B}}\mid f\right)=\sum_{i=2}^{p}D_{KL}^{\mathbf{B},i}\left(f^{\mathbf{B},i}\mid f\right)=$$
$$=\frac{1}{2}\sum_{i=2}^{p}\frac{1}{v_i}\boldsymbol{\delta}_{(i)}^T\mathbf{\Sigma}_{(i-1)}\boldsymbol{\delta}_{(i)}. \qquad (4)$$

Then, the total effect can be expressed as the sum of individual effects and consequently, the global sensitivity analysis can be performed through local analyses of nodes. Some direct results may be useful in applications:

- If all the components in $\boldsymbol{\delta}_{(i)}$ are equal to τ, it follows
$$D_{KL}^{\mathbf{B},i}\left(f^{\mathbf{B},i}\mid f\right)=\frac{\tau^2}{2v_i}\sum_{k=1}^{i-1}\sum_{t=1}^{i-1}\sigma_{tk}$$
- The possibly erroneous arc deletion from node j to node i having $b_{ji}\neq 0$, would yield
$$D_{KL}^{\mathbf{B},i}\left(f^{\mathbf{B},i}\mid f\right)=\frac{1}{2v_i}b_{ji}^2\sigma_{jj}$$
- The effect of arc inclusion from node j to node i introducing b_{ji}^* is also
$$D_{KL}^{\mathbf{B},i}\left(f^{\mathbf{B},i}\mid f\right)=\frac{1}{2v_i}b_{ji}^{*2}\sigma_{jj}$$

These last two cases describe the impact of the conditional and marginal variances for an uncertain knowledge of the exact model giving the distance between DAGs obtained by adding or removing arcs.

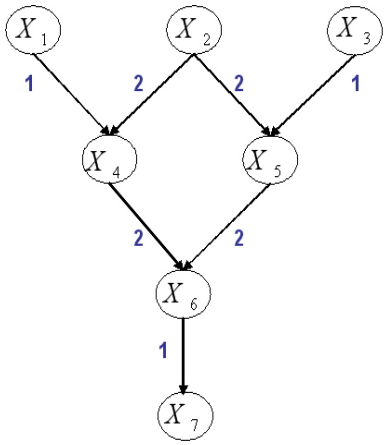


Figure 1: DAG with regression coefficients on the arcs

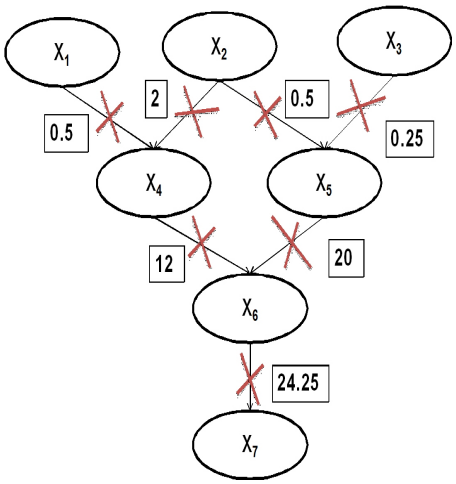


Figure 2: Measure of deviation for each arc removal

Example 1. Let us consider the GBN in Figure1 with parameters:

$$\boldsymbol{\mu} = \mathbf{0}, \mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{D} = diag\,(1, 1, 2, 1, 4, 1, 2)\,.$$

Using that

$$\boldsymbol{\Sigma} = [(\mathbf{I_7} - \mathbf{B})^{-1}]^T \mathbf{D} (\mathbf{I_7} - \mathbf{B})^{-1}$$

the covariance matrix is

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 2 & 2 \\ 0 & 1 & 0 & 2 & 2 & 8 & 8 \\ 0 & 0 & 2 & 0 & 2 & 4 & 4 \\ 1 & 2 & 0 & 6 & 4 & 20 & 20 \\ 0 & 2 & 2 & 4 & 10 & 28 & 28 \\ 2 & 8 & 4 & 20 & 28 & 97 & 97 \\ 2 & 8 & 4 & 20 & 28 & 97 & 99 \end{pmatrix}.$$

The divergences reflecting the effect of each arc removal are shown in Figure 2. It is observed that divergence increases as the depth of the node grows with a maximum in the arc from X_6 to X_7. Therefore, it gives us information about the difference between the original GBN and the particular networks obtained by means of the cancellation of some regression coefficients.

3 The random case

The expression (4) relates easily, global effect to local errors effects due to no interaction. Now, we are going to replace the hypothesis that $\boldsymbol{\Delta}_{\mathbf{B}}$ is known by the assumption that it is a random matrix. The main aim is using the relation (4) to evaluate the impact of uncertainty in $\mathbf{B}$ for a GBN. Both this measure and its value for different nodes can be useful to point the most sensitive nodes so as to compare structures with respect to uncertainty sensitivity.

If we suppose each vector $\boldsymbol{\delta}_{(i)}$ is distributed independent from the remaining errors as $N_{i-1}\,(\mathbf{0}, \mathbf{E}_i)\,, i = 2, ..., p$, while the common value $v_1 = v_2 = \cdots = v_p = v$ is known, each random variable $\boldsymbol{\delta}_{(i)}^T \boldsymbol{\Sigma}_{(i-1)} \boldsymbol{\delta}_{(i)}$ is a quadratic form in normal variables with a chi-squared distribution with $i-1$ degrees of freedom $\chi^2_{(i-1)}$, if and only if $\mathbf{E}_i \boldsymbol{\Sigma}_{(i-1)}$ is a symmetric idempotent matrix of rank $i-1$ (Rencher, 2000).

Also, if $\mathbf{Y}$ is a random vector with mean vector $\boldsymbol{\mu}$, covariance matrix $\boldsymbol{\Sigma}$ and $\mathbf{M}_{p\times p}$ is a non-random matrix, then (Rencher, 2000)

$$E(\mathbf{Y}^T \mathbf{M} \mathbf{Y}) = \boldsymbol{\mu}^T \mathbf{M} \boldsymbol{\mu} + tr(\mathbf{M}\boldsymbol{\Sigma})$$

$$Var(\mathbf{Y}^T\mathbf{M}\mathbf{Y}) = 4\boldsymbol{\mu}^T\mathbf{M}\boldsymbol{\Sigma}\boldsymbol{\mu} + 2tr((\mathbf{M}\boldsymbol{\Sigma})^2). \quad (5)$$

Thus, it results that the mean of the random variable $D_{KL}^{\mathbf{B}}\left(f^{\mathbf{B}} \mid f\right)$ is

$$E\left[D_{KL}^{\mathbf{B}}\left(f^{\mathbf{B}} \mid f\right)\right] = \frac{1}{2v}\sum_{i=2}^{p} tr\left(\Sigma_{(i-1)}\mathbf{E}_i\right).$$

If we express the covariance matrix $\mathbf{E}_i$ in terms of the difference with $\Sigma_{(i-1)}^{-1}$, that is

$$\mathbf{E}_i = \Sigma_{(i-1)}^{-1} + \mathbf{A}_i \ ,$$

then

$$\begin{aligned} tr\left(\Sigma_{(i-1)}\mathbf{E}_i\right) &= tr\left(I_{(i-1)} + \Sigma_{(i-1)}\mathbf{A}_i\right) = \\ &= i - 1 + tr\left(\Sigma_{(i-1)}\mathbf{A}_i\right). \end{aligned}$$

It follows immediately that for a positive semi-definite matrix $\mathbf{A}_i$ —it could be denoted by $\mathbf{E}_i$ "larger than" $\Sigma_{(i-1)}^{-1}$— the minimum mean impact is obtained when $\mathbf{A}_i = \mathbf{0}$ being $k = \frac{1}{2v}\frac{p}{2}(p-1)$ a lower bound for the mean impact of all the matrices $\mathbf{E}_i$ larger than $\Sigma_{(i-1)}^{-1}$. Moreover, imposing $\mathbf{E}_i = \Sigma_{(i-1)}^{-1}$ we could assure each summand is distributed as a $\frac{1}{2v}\chi^2_{(i-1)}$ because $\mathbf{E}_i\boldsymbol{\Sigma}_{(i-1)} = \mathbf{I}_{i-1}$ is a symmetric idempotent matrix of rank $i-1$. The lower bound k can be considered to define the mean relative sensitivity by

$$\frac{E\left[D_{KL}^{\mathbf{B}}\left(f^{\mathbf{B}} \mid f\right)\right]}{k} = \frac{\sum_{i=2}^{p} tr\left(\Sigma_{(i-1)}\mathbf{E}_i\right)}{\frac{p}{2}(p-1)},$$

whenever $\mathbf{E}_i$ "larger than" $\Sigma_{(i-1)}^{-1}$ could be assumed.

3.1 Independent errors

When the hypothesis of independent errors with common variance ϵ^2 can be accepted, that is

$$\mathbf{E}_i = \epsilon^2\mathbf{I}_{(i-1)}, \ i = 2, ..., p \ ,$$

using (5), the quadratic forms of each component have first and second order moments given by:

- $E\left[\boldsymbol{\delta}_{(i)}^T\boldsymbol{\Sigma}_{(i-1)}\boldsymbol{\delta}_{(i)}\right] = \epsilon^2 tr\left(\boldsymbol{\Sigma}_{(i-1)}\right)$
- $Var\left[\boldsymbol{\delta}_{(i)}^T\boldsymbol{\Sigma}_{(i-1)}\boldsymbol{\delta}_{(i)}\right] = 2\epsilon^4 tr[(\boldsymbol{\Sigma}_{(i-1)})^2]$

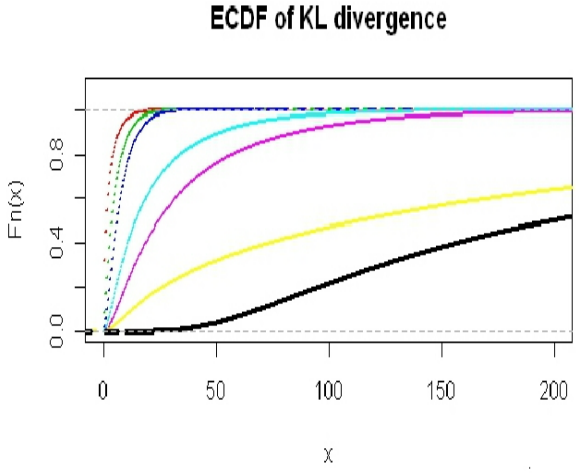


Figure 3: Node divergence from node 2 (*red)* to node 7 *(yellow)* and global divergence *(black)*

Consequently, under the stated conditions, we obtain an increasing average effect according to node depth in the network, independently of the network specification. Figure 3 illustrates the random behavior of Kullback-Leibler divergences displaying the empirical cumulative distribution function (ECDF) for samples from

$$\frac{1}{2v}\boldsymbol{\delta}_{(i)}^T\boldsymbol{\Sigma}_{(i-1)}\boldsymbol{\delta}_{(i)} \ , \ i = 2, ..., 7 \ ,$$

as well as the random global effect for the example discussed above. We have used a simulated sample of size $100,000$, with $v = 1$ and $\epsilon^2 = 5$, for each case.

In this setting, a reasonable procedure to evaluate sensitivity to uncertainty in $\mathbf{B}$ is to analyze the normalized ratio

$$S^{\mathbf{B}}(f) \equiv D_{KL}^{\mathbf{B}}\left(f^{\mathbf{B}} \mid f\right)/\epsilon^2 \ ,$$

that can be interpreted as the distribution variation in terms of the uncertainty variation. Obviously, the random divergence $D_{KL}^{\mathbf{B}}\left(f^{\mathbf{B}} \mid f\right)$ changes with ϵ^2; Figure 4 shows the ECDFs behavior for some ϵ^2 values in Example 1, exhibiting an apparent dominance relation. Nevertheless, the mean as well as the variance of $S^{\mathbf{B}}(f)$ do not depend on ϵ^2; more concretely

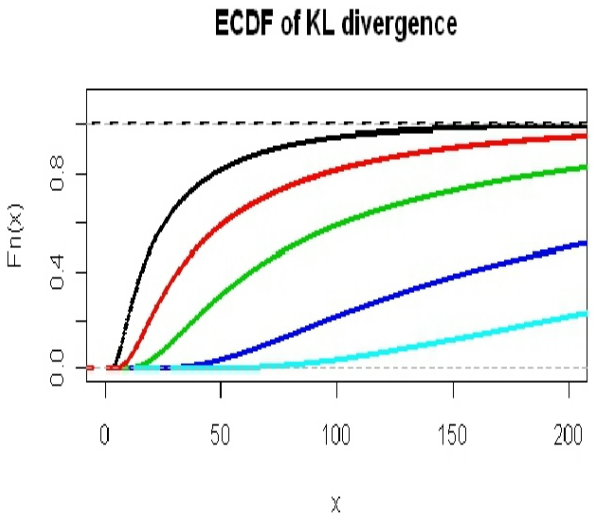


Figure 4: Empirical cumulative distribution function of Kullback-Leibler divergences for GBN in Example 1 with different independent errors: $\varepsilon^2 = 0.5$ *(black)*, 1 *(red)*, 2 *(green)*, 5 *(dark blue)*, 10 *(light blue)*

- $E\left[S^{\mathbf{B}}(f)\right] = \frac{1}{2v}\sum_{i=2}^{p} tr\left(\Sigma_{(i-1)}\right)$
- $Var\left[S^{\mathbf{B}}(f)\right] = \frac{1}{2v^2}\sum_{i=2}^{p} tr\left(\Sigma_{(i-1)}^2\right)$

Relying on the moments invariance we propose to evaluate the GBN sensitivity to uncertainty in $\mathbf{B}$ by

$$E\left[S^{\mathbf{B}}(f)\right] = \frac{1}{2v}\sum_{i=2}^{p} tr\left(\Sigma_{(i-1)}\right) = \\ = \frac{1}{2v}\sum_{i=1}^{p-1}\sigma_{ii}(p-i),$$

where σ_{ii} denotes the variance of X_i. Then, the relative contribution of each node to the total sensitivity measure will be given by

$$\frac{tr\left(\Sigma_{(i-1)}\right)}{\sum_{i=1}^{p-1}\sigma_{ii}(p-i)}. \qquad (6)$$

This result enables us to classify nodes according to greatest contribution to global sensitivity.

Example 2. For the GBN in Example 1 it is obtained

$$E\left[S^{\mathbf{B}}(f)\right] = 63$$

Using (6), the numerical results by nodes are

(2) 0.008, (3) 0.016,
(4) 0.024, (5) 0.071,
(6) 0.12, (7) 0.76

Here the most important values are the contribution of the nodes to the global mean normalized divergence, resulting a significantly large influence for node (7) compared to the rest of nodes in the DAG. Therefore, independent random errors in the regression coefficients of nodes (1) to (5) do not describe very different joint models, however, that is not the case for node (7) and some effort has to be made to bring some additional information to asses the correct regression coefficient value.

4 Conclusions

The factorization of the joint distribution in GBN leads us to an additive decomposition of the Kullback-Leibler divergence. Then, for misspecified regression coefficients, the weight each node has in the global deviation of the initial structure can be determined. Modelling uncertainty with independent random errors provides a highly simplified analysis to achieve an uncertainty sensitivity measure definition that can be easily handled.

Acknowledgments

This research has been supported by the Spanish Ministerio de Ciencia e Innovación, Grant MTM 2008.03282, and partially by GR58/08-A, 910395 - Métodos Bayesianos by the BSCH-UCM from Spain.

References

E. Castillo and U. Kjærulff. 2003. Sensitivity analysis in Gaussian Bayesian networks using a symbolic-numerical technique. *Reliability Engineering and System Safety*, 79:139-148.

M.A. Gómez-Villegas, P. Main, R. Susi. 2007. Sensitivity Analysis in Gaussian Bayesian Networks

Using a Divergence Measure. *Communications in Statistics: Theory and Methods*, 36(3):523–539.

M.A. Gómez-Villegas, P. Main and R. Susi. 2008. Sensitivity of Gaussian Bayesian Networks to Inaccuracies in Their Parameters. *Proceedings of the 4th EuropeanWorkshop on Probabilistic Graphical Models*, pages 145–152.

A. Rencher. 2000. *Linear Models in Statistics*. New York : Wiley.

R. Shachter and C. Kenley. 1989. Gaussian influence diagrams. *Management Science*, 35:527–550.

R. Susi, H. Navarro, P. Main and M.A. Gómez-Villegas. Perturbing the structure in Gaussian Bayesian networks. 2009. *Cuadernos de Trabajo de la E.U. de Estadística*, CT01/2009: 1–21. http://www.ucm.es/info/eue/pagina/cuadernos_trabajo/CT01_2009.pdf.

Causal discovery for linear cyclic models with latent variables

Antti Hyttinen[1], Frederick Eberhardt[2], and Patrik O. Hoyer[1,3]
[1] HIIT / Dept. of Computer Science, University of Helsinki, Finland
[2] Dept. of Philosophy, Washington University in St Louis, MO, USA
[3] CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

We consider the problem of identifying the causal relationships among a set of variables in the presence of both feedback loops and unmeasured confounders. This is a challenging task which, for full identification, typically requires the use of randomized experiments. For linear systems, Eberhardt et al (2010) recently provided a procedure for integrating data from several experiments, and gave a corresponding, but demanding, identifiability condition. In this paper we (i) characterize the underdetermination of the model when the identifiability condition is not fully satisfied, (ii) show that their algorithm is complete with regard to the search space and the assumptions, and (iii) extend the procedure to incorporate the common assumption of faithfulness, and any prior knowledge. The resulting method typically resolves much additional structure and often yields full identification with many fewer experiments. We demonstrate our procedure using simulated data, and apply it to the protein signaling dataset of Sachs et al (2005).

1 Introduction

Researchers are frequently interested in discovering the causal relationships among some given set of variables under study. Such relationships are often represented as directed graphs, in which the variables constitute the nodes of the graph, and a directed edge from one variable x_i to another variable x_j indicates that x_i is a *direct cause* of x_j relative to that set of variables. Since causal relations are not directly observable they must be inferred from available experimental or passive observational data. Several algorithms have been developed that discover as much as possible about such causal relations from passive observational data. One of the difficulties these algorithms confront is the almost inevitable underdetermination of the true causal structure. This problem is exacerbated when there are unmeasured (latent) common causes of the set of variables under consideration, or when there are feedback loops. Consequently, constraints are typically placed on the search space the algorithms consider: The 'FCI' algorithm (Spirtes et al., 2000) only considers *acyclic* causal structures but allows latent variables, while the 'CCD' algorithm of Richardson (1996) can handle cyclic causal systems but does not allow for latents. Even with these restrictions, both algorithms can at best return equivalence classes of causal graphs.

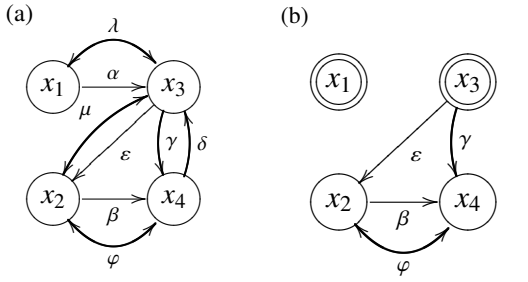
Thus, it is common to turn to experimental data. While randomized experiments break confounding and feedback loops, they pose different challenges. Given that experiments are often costly, how can we identify the causal structure from as few experiments as possible? How can we integrate the data from several existing experiments to yield as much information as possible about the causal relationships among the variables? In this paper, we show how to efficiently perform such causal discovery in *linear* models from a combination of observational and experimental data, while allowing *both* feedback loops and confounding hidden variables.

We consider a standard class of models known as linear non-recursive structural equation models with correlated disturbances (Bollen, 1989). Specifically, let $\mathbf{V} = \{x_1, \ldots, x_N\}$ denote the set of observed variables. Arranging these variables into the vector $\mathbf{x}$, the linear model is given by

$$\mathbf{x} := \mathbf{B}\mathbf{x} + \mathbf{e}, \tag{1}$$

where each element b_{ji} of $\mathbf{B}$ gives the *direct effect* from x_i to x_j, also denoted $b(x_i \rightarrow x_j)$, and

Pp. 153–161 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.



$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \varepsilon & 0 \\ \alpha & 0 & 0 & \delta \\ 0 & \beta & \gamma & 0 \end{pmatrix} \quad \mathbf{B}^m = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \varepsilon & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \beta & \gamma & 0 \end{pmatrix}$$

$$\mathbf{\Sigma_e} = \begin{pmatrix} \sigma_1^2 & 0 & \lambda & 0 \\ 0 & \sigma_2^2 & \mu & \varphi \\ \lambda & \mu & \sigma_3^2 & 0 \\ 0 & \varphi & 0 & \sigma_4^2 \end{pmatrix} \quad \mathbf{\Sigma}_\mathbf{e}^m = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & \varphi \\ 0 & 0 & 1 & 0 \\ 0 & \varphi & 0 & \sigma_4^2 \end{pmatrix}$$

Figure 1: (a) Example model. (b) Manipulated model, corresponding to an experiment $(\mathbf{J}_m, \mathbf{U}_m)$ where $\mathbf{J}_m = \{x_1, x_3\}$ and $\mathbf{U}_m = \{x_2, x_4\}$. Disturbance variables are not shown.

the random vector $\mathbf{e}$ contains zero-mean *disturbance* (*error*) variables with a covariance matrix $\mathbf{\Sigma_e} = E\{\mathbf{e}\mathbf{e}^T\}$. An example model is given in Figure 1a.

An experiment $\mathcal{E}_m = (\mathbf{J}_m, \mathbf{U}_m)$ divides $\mathbf{V}$ into two mutually exclusive and exhaustive sets $\mathbf{J}_m$ and $\mathbf{U}_m$. $\mathbf{J}_m$ contains the variables subject to an intervention in $\mathcal{E}_m$ and $\mathbf{U}_m$ contains the variables that are passively observed in that experiment. In such an experiment, all variables $x_i \in \mathbf{J}_m$ are independently and simultaneously randomized. In terms of the directed graph, this is represented by cutting all edges *into* any such variable (Pearl, 2000). In terms of the parameters, we thus have a *manipulated* model $(\mathbf{B}^m, \mathbf{\Sigma}_\mathbf{e}^m)$, where $\mathbf{B}^m$ is equal to $\mathbf{B}$ except that all rows corresponding to such randomized variables are set to zero, and $\mathbf{\Sigma}_\mathbf{e}^m$ equals $\mathbf{\Sigma_e}$ but with all rows and columns corresponding to randomized variables set to zero, except for the corresponding diagonal element which is set to equal one due to the fixed variance of the randomization. See Figure 1b.

If the variables cannot be ordered such that the corresponding $\mathbf{B}$ is lower-triangular we have a truly non-recursive system that *cannot* be represented as a directed *acyclic* graph (DAG). If $\mathbf{\Sigma_e}$ has non-zero off-diagonal entries the system is said to exhibit confounding due to latent variables. In each experiment $\mathcal{E}_m$ the data are generated such that a random sample of disturbance vectors $\mathbf{e}$ are drawn with (manipulated) covariance $\mathbf{\Sigma}_\mathbf{e}^m$, and we observe the vectors $\mathbf{x}$ (and hence their covariance $\mathbf{\Sigma}_\mathbf{x}^m$) generated (at equilibrium) from the model with (manipulated) coefficient matrix $\mathbf{B}^m$. For the feedback system to reach equilibrium, the absolute values of all eigenvalues of $\mathbf{B}^m$ must be smaller than one.[1] A passive observational dataset is obtained in an 'experiment' in which $\mathbf{J}_m = \emptyset$ and $\mathbf{U}_m = \mathbf{V}$.

In an experiment in which $x_i \in \mathbf{J}_m$ and $x_j \in \mathbf{U}_m$, the *experimental effect* of x_i on x_j, denoted $t(x_i \rightsquigarrow x_j \,||\, \mathbf{J}_m)$, is defined as the covariance of x_i and x_j in this experiment, i.e. $\mathbf{\Sigma}_\mathbf{x}^m[i, j]$. This is equal to the sum of the strenghts of all uncut directed paths from x_i to x_j, where the strength of a path is simply the product of the edge coefficients (direct effects) on that path.[2]

Our task is to devise a sequence of experiments $(\mathcal{E}_1, \ldots, \mathcal{E}_M)$, and corresponding estimation procedure, that fully identifies the parameter matrices $\mathbf{B}$ and $\mathbf{\Sigma_e}$, in the sense that the estimates are consistent (converge to the true values in the infinite sample limit). Alternatively or in addition, for a fixed set of experiments one would like to recover as many as possible of these parameters. Note that if all but one variable is randomized in an experiment (i.e. $\mathbf{J}_m = \mathbf{V} \setminus \{x_j\}$) one can consistently estimate all direct effects $b(x_i \rightarrow x_j)$, $\forall i \neq j$, since in this experiment the direct effects equal the experimental effects. Thus one solution to identify $\mathbf{B}$ consists of $M = N$ such experiments each intervening on $N - 1$ variables. If in addition a passive observational dataset were available, one can obtain a consistent estimate of $\mathbf{\Sigma_e}$ from the identity $\mathbf{\Sigma_x} = (\mathbf{I} - \mathbf{B})^{-1}\mathbf{\Sigma_e}(\mathbf{I} - \mathbf{B})^{-T}$, where $\mathbf{\Sigma_x}$ is the covariance of $\mathbf{x}$ in a passive observational dataset.

Can we get by with fewer experiments? Recently, Eberhardt et al (2010) provided a procedure that identifies the full matrix $\mathbf{B}$ if and only if the following *pair condition* holds for each ordered variable pair $(x_i, x_j) \in \mathbf{V} \times \mathbf{V}$, with $i \neq j$: there is an experiment $\mathcal{E}_m = (\mathbf{J}_m, \mathbf{U}_m)$ in the sequence in which $x_i \in \mathbf{J}_m$ and $x_j \in \mathbf{U}_m$.

[1] As in (Eberhardt et al., 2010) we assume that this condition is satisfied for all possible manipulations of the $\mathbf{B}$-matrix.

[2] Note that this sum has an infinite number of terms when the model is cyclic.

However, several questions were left unanswered in their study. First, if the pair condition is not satisfied for all ordered pairs, which direct effects are identified and which are not? Second, is it possible that some alternative procedure might identify the full model even when for some pairs the condition is not satisfied? Finally, satisfying the pair condition for all ordered pairs is a very high bar for the identifiability of the underlying causal structure, as it requires that each variable must be subject to at least one intervention at some point in the sequence of experiments. For any observed variable that is not subject to an intervention their algorithm can only discover the causal structure marginalized over that variable. Thus, can we make use of prior knowledge when available, or strengthen some of the assumptions, to avoid requiring the pair condition for all pairs? We answer these three questions in Sections 2–4, respectively. Then, in Section 5, we describe a simple adaptive procedure for selecting the sequence of experiments, while providing simulations in Section 6 and an application to the protein signaling dataset of Sachs et al (2005) in Section 7. Conclusions are given in Section 8.

2 Characterization of underdetermination

Eberhardt et al (2010) showed that if the pair condition (see Section 1) is not satisfied for all ordered pairs then their estimation procedure leaves some total effects undetermined, and hence some elements of the direct effects matrix $\mathbf{B}$ are undetermined as well. They then suggested a numerical heuristic to identify the set of edges that are not yet determined. Here we show how, using an alternative formulation of the procedure, we obtain a characterization of the remaining underdetermination in the direct effects.

From an experiment $\mathcal{E}_m = (\mathbf{J}_m, \mathbf{U}_m)$, with $x_i \in \mathbf{J}_m$ and $x_j \in \mathbf{U}_m$, Eberhardt et at (2010) showed that one can derive linear constraints on the *total effects* entailed by the model. For the purposes of the present paper, it is much more useful to work with the *direct effects*. We can similarly derive the following linear constraints expressing the experimental effects as a linear sum of direct effects:

$$t(x_i \rightsquigarrow x_j \,||\, \mathbf{J}_m) = \sum_{x_k \in \mathbf{U}_m \setminus x_j} t(x_i \rightsquigarrow x_k \,||\, \mathbf{J}_m)\, b(x_k \to x_j) + b(x_i \to x_j) \quad (2)$$

For instance, for the experiment of Figure 1b, with $i = 3$ and $j = 4$, we get $t(x_3 \rightsquigarrow x_4 \,||\, \mathbf{J}_m) = t(x_3 \rightsquigarrow x_2 \,||\, \mathbf{J}_m) b(x_2 \to x_4) + b(x_3 \to x_4)$, which is easily verified. This equation holds for cyclic as well as acyclic systems, and derives from the definition of the experimental effect from x_i to x_j (see Section 1). When grouping all directed paths from x_i to x_j according to the final edge into x_j, each such group represents another experimental effect obtainable from $\mathcal{E}_m$. Note that the experimental effects $t(x_i \rightsquigarrow x_j \,||\, \mathbf{J}_m)$ and $t(x_i \rightsquigarrow x_k \,||\, \mathbf{J}_m)$ are numerical quantities estimated from the experiments, and the unknowns are the direct effects $b(x_k \to x_j)$, $\forall x_k \in \mathbf{U}_m \setminus x_j$, and $b(x_i \to x_j)$.

This alternative representation immediately lends itself to the identification of the underdetermination in the direct effects. All linear equations of the form of equation 2 can be written into a matrix equation $\mathbf{Kb} = \mathbf{k}$, where the unknown vector $\mathbf{b}$ groups the elements of the unknown matrix $\mathbf{B}$. A given element of $\mathbf{b}$ (and hence of $\mathbf{B}$) is undetermined if and only if that element is involved in the nullspace of the constraint matrix $\mathbf{K}$.

The above characterization does not provide much of an understanding of the underdetermination in terms of the graph structure. Nevertheless, consider the following. Any direct effects $b(\bullet \to x_j)$ into x_j are only constrained by experiments in which $x_j \in \mathbf{U}_m$. That is, the direct effects occur only in constraints of this type and there is only one (linearly independent) such constraint for each ordered pair $(\bullet, x_j)$ that the pair condition is satisfied for. Thus, in the general case, when the pair condition is not satisfied for a particular pair (x_i, x_j) then the entire j:th row of $\mathbf{B}$ is undetermined. Conversely, since the direct effects into x_j are the only direct effects that enter into these types of constraints, it follows that if the pair condition is satisfied for all pairs $(\bullet, x_j)$, then $n-1$ constraints can be determined and the row in $\mathbf{B}$ specifying the direct effects into x_j is fully identified. Hence, to *guarantee* the identifiability of a given direct effect $b(x_i \to x_j)$, it is necessary to satisfy the pair condition for all ordered pairs (x_k, x_j) with $k \neq j$. Note that in particular

graphs it may be possible to identify a direct effect $b(x_i \rightarrow x_j)$ even when the above condition is not true. In all cases, our code package provides the user with an explicit characterization of which coefficients are determined and which are not, given the results of any provided set of experiments.

3 Completeness of the procedure

An important question concerns whether the procedure introduced by Eberhardt et al (2010) fully exploits all the available data. Each experiment $\mathcal{E}_m = (\mathbf{J}_m, \mathbf{U}_m)$ supplies a data covariance matrix $\mathbf{\Sigma}^m_{\mathbf{x}}$, in which each entry $\mathbf{\Sigma}^m_{\mathbf{x}}[i, j]$ specifies the covariance between x_i and x_j in the experiment $\mathcal{E}_m$. The procedure as described (and the related pair condition theorem the authors gave) is based exclusively on constraints due to the experimental effects $t(x_i \rightsquigarrow x_j \,||\, \mathbf{J}_m) = \mathbf{\Sigma}^m_{\mathbf{x}}[i, j]$ where $x_i \in \mathbf{J}_m$ and $x_j \in \mathbf{U}_m$. These covariances only constitute part of the information contained in a data convariance matrix $\mathbf{\Sigma}^m_{\mathbf{x}}$. In particular, the covariances between non-intervened variables, $\mathbf{\Sigma}^m_{\mathbf{x}}[j, k]$ with $x_j, x_k \in \mathbf{U}_m$, were not utilized at all.[3] It is tempting to think that this additional source of information could provide further leverage to identify the causal structure, and thereby reduce the demands for identifiability. However, we have the following negative result:

Lemma 1. *Let the true model generating the data be* $(\mathbf{B}, \mathbf{\Sigma_e})$. *For each of the experiments* $(\mathcal{E}_m)_{m=1,...,M}$ *the obtained data covariance matrix is* $\mathbf{\Sigma}^m_{\mathbf{x}}$. *If there is a direct effects matrix* $\widehat{\mathbf{B}} \neq \mathbf{B}$ *such that for all* $(\mathcal{E}_m)_{m=1,...,M}$ *and all* $x_i \in \mathbf{J}_m$ *and* $x_j \in \mathbf{U}_m$ *it produces the same experimental effects* $t(x_i \rightsquigarrow x_j \,||\, \mathbf{J}_m)$, *then the model* $(\widehat{\mathbf{B}}, \widehat{\mathbf{\Sigma}}_{\mathbf{e}})$ *with* $\widehat{\mathbf{\Sigma}}_{\mathbf{e}} = (\mathbf{I} - \widehat{\mathbf{B}})(\mathbf{I} - \mathbf{B})^{-1}\mathbf{\Sigma_e}(\mathbf{I} - \mathbf{B})^{-T}(\mathbf{I} - \widehat{\mathbf{B}})^T$ *has data covariance matrices* $\widehat{\mathbf{\Sigma}}^m_{\mathbf{x}} = \mathbf{\Sigma}^m_{\mathbf{x}}$ *for all* $m = 1, ..., M$.

Proof. The proofs for all results given in this paper are provided in online supplementary material at: `http://cs.helsinki.fi/u/ajhyttin/exp/`

When $\mathbf{B}$ is underdetermined Lemma 1 constitutes a constructive proof that any measure of the covariance between two non-intervened variables provides no additional help with the identifiability of $\mathbf{B}$ in the model space considered in (Eberhardt et al., 2010). Intuitively, this result is a consequence of the dependence of the covariances between non-intervened variables on the model's disturbance covariance matrix $\mathbf{\Sigma_e}$. The additional $(n^2 + n)/2$ unknown parameters of $\mathbf{\Sigma_e}$ swamp the gains these covariance measures provide. Lemma 1 implies that the pair condition theorem can be strengthened to state that the method of Eberhardt et al. (2010) is complete with regard to the information contained in the data covariance matrices for the search space they consider.

Theorem 1 (Completeness Theorem)**.** *Given the data covariance matrices from a sequence of experiments* $(\mathcal{E}_m)_{m=1,...,M}$ *over the variables in* $\mathbf{V}$, *all direct effects* $b(x_i \rightarrow x_j)$ *are identified if and only if the pair condition is satisfied for all ordered pairs of variables w.r.t. these experiments.*[4]

However, measures of the covariances between non-intervened variables *are necessary* to identify the disturbance covariance matrix $\mathbf{\Sigma_e}$ (specifying the latent variables). In the original procedure $\mathbf{\Sigma_e}$ was determined using measurements from an additional passive observational dataset (with $\mathbf{J}_m = \emptyset$). It can be shown that a much weaker condition, similar to the pair condition for experimental effects, is necessary and sufficient for the identification of $\mathbf{\Sigma_e}$, if $\mathbf{B}$ is already determined. We can thus state the following general theorem of model identifiability:

Theorem 2 (Model Identifiability Theorem)**.** *Given a sequence of experiments* $(\mathcal{E}_m)_{m=1,...,M}$ *over the variables in* $\mathbf{V}$ *the model* $(\mathbf{B}, \mathbf{\Sigma_e})$ *is fully identified if and only if for each ordered pair of variables* (x_i, x_j) *there is an experiment* $\mathcal{E}_b = (\mathbf{J}_b, \mathbf{U}_b)$ *with* $x_i \in \mathbf{J}_b$ *and* $x_j \in \mathbf{U}_b$ *and another experiment* $\mathcal{E}_e = (\mathbf{J}_e, \mathbf{U}_e)$ *with* $x_i, x_j \in \mathbf{U}_e$.

Thus, the good news is that the algorithm given by (Eberhardt et al., 2010) does as well as it possibly could with regard to identifiability. The bad news is that the generality of its search space implies that the conditions for identifiability are very demanding. Hence, in the following section we consider how the use of background knowledge or an additional assumption of faithfulness can help.

[3] Obviously, when two variables are both in $\mathbf{J}_m$, then the covariance between them in that experiment is zero by assumption, since simultaneous interventions are assumed to make the intervened variables independent.

[4] Note the inevitable limitation of identifiability with regard to self-loops discussed in (Eberhardt et al., 2010).

4 The faithfulness assumption

When two variables are independent one commonly assumes that they are not causally connected. However, this assumption is non-trivial, since it precludes, for example, cases where two variables are connected by two separate pathways that exactly cancel each other out. The two variables are then probabilistically independent, while they are still causally connected by a directed path.

For instance, in the model of Figure 2a, in an experiment where x_1 is randomized, and x_2 and x_3 are (passively) observed, x_1 and x_3 would be found marginally independent, but dependent conditional on x_2. Such an observation could have many possible alternative explanations, two of which are shown in Figures 2b and 2c. In such cases scientists do often make the assumption that an absence of a correlation is an indication of the absence of a causal connection, and hence favor explanations (b) and (c) over (a). In this section we introduce inference rules that take advantage of this intuition.

The structure of a model entails certain marginal and conditional independencies in the resulting distribution; these are characterized by the *Markov condition*, and Spirtes (1995) has shown that the familiar concept of d-separation specifies all and only the independencies entailed in all linear structural equation models, including cyclic (non-recursive) models. The intuition given above is then formalized in the *faithfulness* assumption, which states that all independencies in the population distribution are derived from the structure of the graph, rather than specific parameter values (Spirtes et al., 2000; Pearl, 2000).

For maximum generality, the algorithm in (Eberhardt et al., 2010) did not use the assumption of faithfulness. However, given the demanding identifiability conditions (see Section 1), it is worth investigating whether faithfulness might add substantial benefit when the pair condition is not satisfied for all ordered pairs of variables.

In general, causal discovery based on faithfulness proceeds in two steps. First, independence tests are used to detect the absence of edges between pairs of variables. Subsequently, the detected absences are used to 'orient' as many as possible of the remaining edges. We here employ an analogous approach.

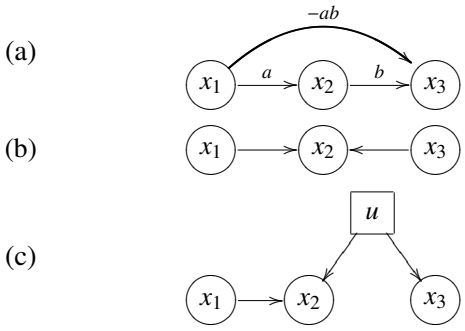


Figure 2: Example graphs. In (c) u is unmeasured.

First, we rely on the fact that if, in any experiment $\mathcal{E}_m = (\mathbf{J}_m, \mathbf{U}_m)$, two non-intervened variables $x_i, x_j \in \mathbf{U}_m$ are marginally or conditionally independent (with any conditioning set not including x_i and x_j), then by faithfulness $b(x_i \to x_j) = b(x_j \to x_i) = \mathbf{\Sigma_e}[i, j] = 0$. Similarly, if $x_i \in \mathbf{J}_m$ and $x_j \in \mathbf{U}_m$ are found marginally or conditionally independent, faithfulness requires that $b(x_i \to x_j) = 0$. (Note that while independencies imply the absence of edges, dependencies do not necessarily imply the presence of any edge between a given pair of edges.) In our implementation we run the statistically and computationally efficient schedule of independence tests suggested by the PC-algorithm (Spirtes et al., 2000) on the data from each experiment separately. Althought PC is designed for a different search space, any indepencies found are usable in our procedure as well. Any obtained constraints are termed *skeleton constraints*.

The orientation rules of the second step of the inference are more intricate. We cannot simply adopt the orientation rules from existing constraint-based algorithms since they only provide orientation rules for search spaces where the true causal structure either contains latent variables but no cycles (FCI, (Spirtes et al., 2000)) or contains cycles but no latent variables (CCD, (Richardson, 1996)). Since our model space contains both latent variables and cycles, and we have the advantage of experiments, different orientation rules are required. We employ the following two rules that take advantage of the orientation supplied by interventions. Any constraints thus obtained are termed *orientation constraints*:

(1) If in a given experiment we have $x_i \in \mathbf{J}_m$ and $x_j, x_k \in \mathbf{U}_m$, and $t(x_i \rightsquigarrow x_j \,||\, \mathbf{J}_m) \neq 0$ but $t(x_i \rightsquigarrow x_k \,||\, \mathbf{J}_m) = 0$, then $b(x_j \to x_k) = 0$ by faithful-

ness. For instance, in the model of Figure 1a, in an experiment with $\mathbf{J}_m = \{x_3\}$, we see an experimental effect from x_3 to x_2, but no experimental effect from x_3 to x_1. We would thus infer that $b(x_2 \to x_1) = 0$. Similarly we would infer that $b(x_4 \to x_1) = 0$. This rule is sound because by the antecedent there is a directed path from x_i to x_j so that, were there a non-zero direct effect $b(x_j \to x_k)$ it would follow that there would be a directed path from x_i to x_k, which for a faithful model would imply a non-zero experimental effect of x_i on x_k.

(2) Again, if we have $x_i \in \mathbf{J}_m$ and $x_j, x_k \in \mathbf{U}_m$, and observe $t(x_i \rightsquigarrow x_j \,||\, \mathbf{J}_m) \neq 0$, and in addition x_i is conditionally independent of x_k given x_j, then we infer that $b(x_k \to x_j) = 0$ and $\mathbf{\Sigma_e}[j,k] = 0$. The rule is correct under faithfulness because we must have a directed path from x_i to x_j, so if there existed a direct effect from x_k to x_j (or a confounder between the two) by faithfulness this would cause a dependence between x_i and x_k when conditioned on the collider x_j.

Since all the new constraints are (trivially) linear in the direct effects, they can be directly added to the set of constraints on the direct effects given by the experimental effects described in Section 2. The combined system can then be solved and the underdetermination characterized as before.

We note that the above rules clearly do not exhaust the inferences that could (potentially) be drawn by faithfulness. It is an open (and intriguing!) problem to devise a set of *complete* rules for causally insufficient, cyclic discovery.

Finally, if by domain knowledge we are guaranteed that x_i does not have a direct effect on x_j (with respect to $\mathbf{V}$), then we may naturally add the constraint $b(x_i \to x_j) = 0$. Such prior knowledge may be particularly useful for dense graphs or models which are close to unfaithful, when the faithfulness rules would not apply or would be unreliable.

5 Adaptive selection of experiments

While the form of the constraints obtained from the experimental effects (given in Section 2) can be predicted ahead of performing the experiments, constraints due to faithfulness come as an unexpected 'bonus': We cannot know ahead of time which independencies will be uncovered. Hence, to minimize the total number of experiments, one must react and adapt the sequence to newly discovered constraints.

We have found that a simple greedy selection procedure works well. As in the original procedure of Eberhardt et al (2010), we keep a list of which ordered pairs have the pair condition satisfied. However, in addition to pairs satisfied purely on the basis of the choice of previous experiments $\mathcal{E}_m$, we also treat any pair (x_i, x_j) as if it is satisfied whenever, using the characterization of underdetermination in Section 2, the coefficient $b(x_i \to x_j)$ is determined. This includes both coefficients directly determined by background knowledge or our faithfulness rules, as well as coefficients indirectly determined by the collection of all existing constraints. The next experiment is selected such that we maximize the number of ordered pairs for which the pair condition is guaranteed to be satisfied after the experiment, arbitrarily breaking ties. In the following simulations, we demonstrate that, for sparse graphs, this is an effective selection protocol.

6 Simulations

In this section, we describe a set of simulations on random graphs that we used to investigate the power provided by the faithfulness assumption.[5]

We generated a large number of random graphs over 10 variables, with sparsity ranging from zero edges up to 60 edges (out of 135 possible, counting both direct and confounding edges). The coefficients were drawn uniformly from $[0.3, 0.8]$ with random sign, and stability was examined by checking the eigenvalues of the resulting $\mathbf{B}$.

First, we study the theoretical limit behavior (infinite sample limit) of our procedure. In Figure 3a, we plot the average number of experiments needed to completely identify the model, as a function of the underlying model sparsity and the number of interventions per experiment. For one intervention per experiment (left panel), in the absence of faithfulness rules the full 10 experiments are needed regardless of sparsity, while for relatively sparse graphs on average a few experiments can be saved by utilizing faithfulness. When intervening on three variables per experiment (right panel), 7 experiments

[5] We encourage the interested reader to try out the method. A complete implementation (reproducing all the simulations) is available at: `http://cs.helsinki.fi/u/ajhyttin/exp/`

are needed in the basic case to satisfy the pair condition for all pairs, and significant savings can be obtained when using the faithfulness assumption. Meanwhile, Figure 3b shows the number of ordered pairs (a lower bound of the rank of the constraint matrix) satisfied after only three experiments, as a function of sparsity. It can be seen that for sparse graphs, most of the structure of the graph has already been discovered at this stage of the sequence of experiments.

Second, we look at finite sample behavior. Figure 4 shows the number of experiments used, as well as the resulting accuracy (linear correlation between estimated and true coefficients). In each experiment, 10,000 samples were used. We note the following: To guarantee high accuracy in dense graphs, the pair condition must be satisfied for all pairs based on the experimental setup alone (as in the 'no faithfulness' procedure). However, when the true model is sparse, significant savings in terms of the number of experiments are possible. Especially when intervening on several variables in each experiment, the full model is typically identified with high accuracy in just 4 experiments. Accuracy drops markedly for dense graphs, as the number and size of possible conditioning sets is so large that inevitably some dependent variables are mistakenly inferred to be independent, yielding large errors. These erroneous inferences cause the number of experiments to stay roughly constant as a function of the number of edges in the graph, in marked contrast to the infinite sample limit of Figure 3a.

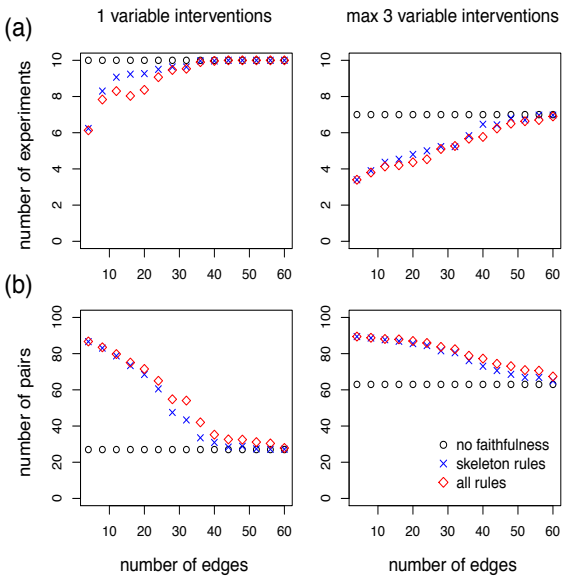


Figure 3: Performance of the procedure in the infinite-sample limit. (a) Number of experiments needed to identify the full model, and (b) amount of structure discovered after three experiments, as a function of the number of edges in the graph.

7 Application to flow cytometry data

Finally, we applied the algorithm (with the faithfulness rules) to the flow cytometry data of Sachs et al (2005). In this data set only 4 of the 11 measured variables were manipulated with no changes made to the background conditions, see (Eberhardt et al, 2010) for details. This meant that the pair condition was satisfied for only 40 ordered pairs out of the total of 110. Together with the faithfulness rules, however, these experiments were enough to determine a majority of the direct effects in the model.

We ran the inference procedure with a variety of parameter settings (significance threshold for statistical dependence, using the full faithfulness rules or only the skeleton rules, threshold for detecting determined vs undetermined coefficients, etc). A typical result is shown in Figure 5. We emphasize that our method assumes linearity, while the true model is likely to be at least somewhat non-linear. Thus the main interest lies in the resulting structure, and possibly the signs of the direct effects. While there were differences in the inferred graphs, many features were common to all of our results.

In particular, we always find (a) the well known Raf→Mek→Erk pathway (and invariably, in addition, Mek seems to have a direct effect on Raf), (b) PKC influences (directly or indirectly) a number of targets, including Raf, PKA, and Jnk, and (c) a strong association between PIP2 and PIP3 (and these are sometimes though not always connected with Plcg). These features are quite compatible with the 'ground truth' (from the literature) model given by Sachs et al (2005). However, our procedure also suggests that many of the variables have effects *into* PKA, something not supported by their model. Finally, we note that our method quite often detects bidirectional relationships; at this point, we do not

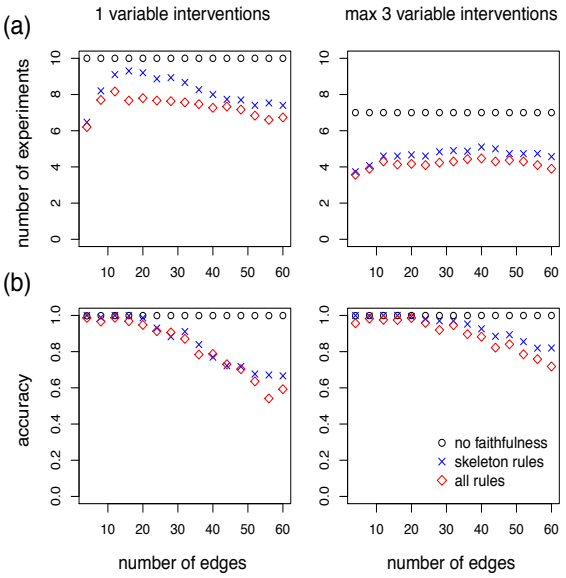


Figure 4: Results on simulated (sample) data. (a) Number of experiments needed to identify the model, and (b) accuracy (correlation between the estimates and the true values), as a function of the number of edges in the graph.

know whether this is due to the nature of our procedure or whether this is a true feature of the data.

8 Conclusions

The discovery procedure is relatively unique in the generality of the model space considered. While there exists a large body of work on learning *acyclic* causal structures with or without hidden variables, there is comparatively little on learning models involving feedback loops. Richardson (1996) gave a constraint-based discovery procedure for passive observational data, but did not allow for latent variables. More recently, both Schmidt and Murphy (2009) and Itani et al (2010) have introduced probabilistic models for cyclic structures involving discrete-valued variables, and given related discovery procedures. While all of these methods use somewhat different models and assumptions, ultimately they nevertheless all share the goal of elucidating causal structure among variables that are recurrently connected. A thorough empirical study, comparing the various methods both on simulations and on a number of real datasets, would be an important next step.

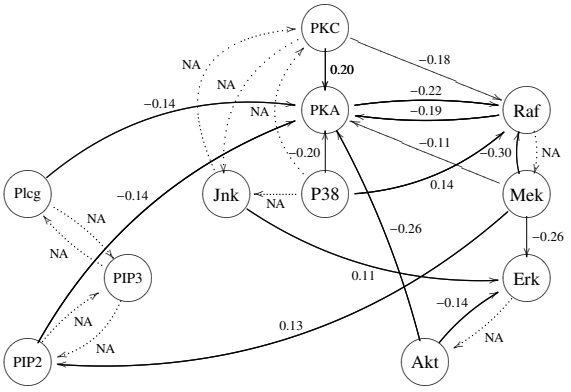


Figure 5: Protein interaction graph inferred from the dataset of Sachs et al (2005), with 'NA' denoting non-identified edge strengths (which could potentially be zero, hence these edges are plotted with dotted lines). Settings: significance threshold 0.05, only skeleton rules.

Acknowledgments

A.H. and P.O.H. were funded by Univ. of Helsinki Research Funds and the Academy of Finland.

References

K. A. Bollen. 1989. *Structural Equations with Latent Variables*. John Wiley & Sons.

F. Eberhardt, P. O. Hoyer, and R. Scheines. 2010. Combining experiments to discover linear cyclic models with latent variables. In *AISTATS 2010*.

S. Itani, M. Ohannessian, K. Sachs, G. P. Nolan, and M. A. Dahleh. 2010. Structure learning in causal cyclic networks. In *JMLR W&CP*, volume 6, pages 165–176.

J. Pearl. 2000. *Causality*. Oxford University Press.

T. Richardson. 1996. *Feedback Models: Interpretation and Discovery*. Ph.D. thesis, Carnegie Mellon.

K. Sachs, O. Perez, D. Pe'er, D.A. Lauffenburger, and G.P. Nolan. 2005. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

M. Schmidt and K. Murphy. 2009. Modeling discrete interventional data using directed cyclic graphical models. In *UAI '09*.

P. Spirtes, C. Glymour, and R. Scheines. 2000. *Causation, Prediction and Search*. MIT Press, 2 edition.

P. Spirtes. 1995. Directed cyclic graphical representation of feedback models. In *UAI'95*.

Information enhancement for approximate representation of optimal strategies from influence diagrams

Finn Verner Jensen
Aalborg University, Denmark
fvj@cs.aau.dk

Elena Gatti
Università di Milano-Bicocca, Italy
elena.gatti@disco.unimib.it

Abstract

The main source of complexity problems for large influence diagrams is that the last decisions have intractably large spaces of past information. Usually, it is not a problem when you reach the last decisions; but when calculating optimal policies for the first decisions, you have to consider all possible future information scenarios. This is *the curse of knowing that you shall not forget*. The usual approach for addressing this problem is to reduce the information through assuming that you do forget something (LIMID, (Nilsson and Lauritzen, 2001)), or to abstract the information through introducing new nodes (Jensen, 2008). This paper takes the opposite approach, namely to assume that you know more in the future than you actually will. We call the approach *information enhancement*. We reduce the future information scenarios by adding information links. We present a systematic way of determining information links to add.

1 Introduction

As opposed to decision trees, influence diagrams are easy to enter to a computer. Hence, the hard job is to establish a solution: a set of optimal policies $\{\delta_i\}$, one for each decision D_i. There are several algorithms for solving IDs (Olmsted, 1983) (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 with the aim of forecasting the expected utility. The way it is performed is through *variable elimination*: all variables are successively removed from the graph, and when a variable A is removed, the resulting graph will hold a link between any pair of A's neighbors. For IDs the elimination order has to respect the reverse (partial) temporal ordering induced by the structure of the ID. We assume the reader to be familiar with standard concepts and methods for probabilistic graphical models (d-separation, triangulation, junction trees).

The solution phase may be very demanding with respect to time and space, but it is an off-line activity where you are not bound by tough resource constraints. The complexity problem arises when you eliminate a variable A, and you have to work with a joint table over an almost too large set of neighbors of A.

The next task is to *represent* the solution. The policies in the solution may have very large domains. Take for example the last decision in a sequence of ten. Then the policy δ_{10} is a function whose domain may include all previous observations and decisions.

For illustration, look at Figure 1. The domain for δ_4 contains 11 variables. This means that variable elimination will have to deal with tables with 10^{11} entries. It is an off-line activity, and

Pp. 161–169 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

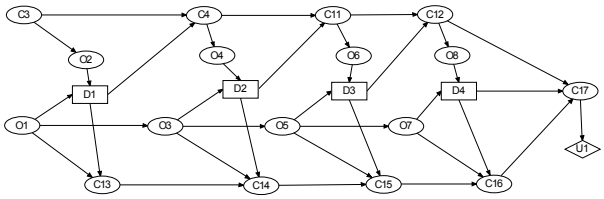


Figure 1: An influence diagram over variables with ten states.

you may succeed by spending much time, space and exploit sophisticated machines and/or cloud computing. Although it may seem intractable to represent δ_4 for fast online access, it is not a problem: the ID itself is a very compact representation of a policy for the last decision. When you have to take the decision D_4, you know the state of the information variables, and it is an easy computational task to find an optimal decision.

The problem concerns the first decision. When taking the first decision you must anticipate what you will do when taking the last decision. However, you do not know the information available at that time, and therefore you in principle have to work with the joint probability of all the unknown information variables (including future decisions). This is what we call *the curse of knowing that you shall not forget*.

We consider a solution of an influence diagram as a representation of a set of policies. The policy for the first decisions may be represented as a look-up table and the policies for the last decisions may be represented as influence diagrams. Usually, the domain of the first decision is not extremely large, so you may off-line compute an optimal policy, which can be stored for fast access. We shall address the decisions in between, and we construct influence diagram representations, where policies of future decisions are approximated through reduction of their domain (see Figure 2 for an illustration).

If the ID in Figure 2 is used to represent the policy δ_7, then the nodes P_1 to P_6 are known. That is, the state of these nodes are entered before the solution algorithm is started, and they do not contribute to the space complexity of the solution algorithm. The problem for the situation in Figure 2 is twofold; the space of the past for D_7 is too large such that δ_7 cannot be represented as a look-up table, and the space of future information relevant for D_{10} is so large that an on-line solution of the ID is not tractable.

The problem has previously been addressed by an approach, which can be characterized as *information abstraction*: you aim at determining a small set of variables which serve as an abstraction of the actual information. This may done with the LIMID approach (Nilsson and Lauritzen, 2001), where it is assumed that some information will be forgot in the future, or it may be done through introduction of new nodes (like history nodes) through which the information is passed (Jensen, 2008).

In this paper we take the opposite approach, which we call *information enhancement*: we assume the decision maker to be more informed than actually will be the case.

2 Information enhancement

Our information enhancement approach consists of determining a small set of variables, which if known would overwrite the actual information. We shall use the terms *disclosed* and *closed* for variables with known state and unknown state, respectively.

The idea behind information enhancement is to find a cut set $\mathcal{S}$ which d-separates the rest of the information from the relevant utilities. When $\mathcal{S}$ has been determined, we assume it to be disclosed when taking the future decision. We shall say that the new information nodes are *enhanced*

To illustrate the approach, consider a finite horizon partially observable Markov decision process (POMDP) (Drake, 1962) (see Figure 3).

As the nodes C_1 to C_7 may be a compound of several variables, and the observed nodes may also be a set of variables, we may assume that all the chance variables have 50 states. Now, consider the decision D_3. The past is too large for a direct representation of δ_3, and the influence diagram with the past of D_3 instantiated is also too complex. We can approximate δ_3

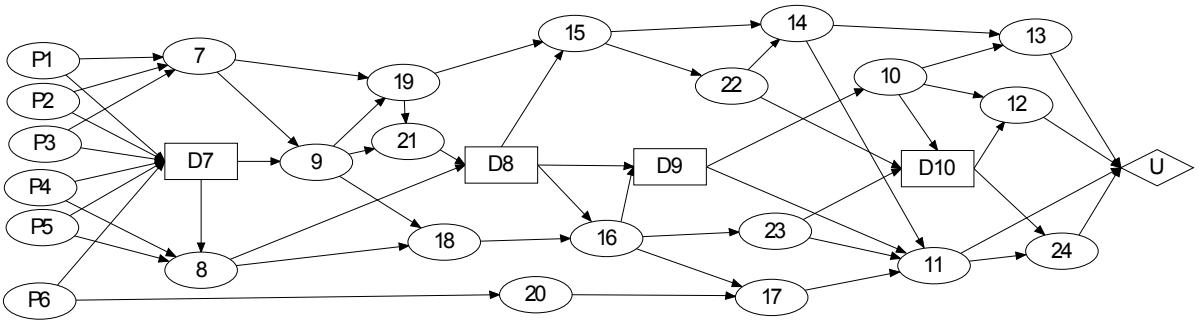


Figure 2: The general situation. You are in the middle of a series of decisions (D_7); you have collected much information (P_1 to P_6), and in order to determine an optimal decision for D_7, you have to anticipate a future decision (D_{10}).

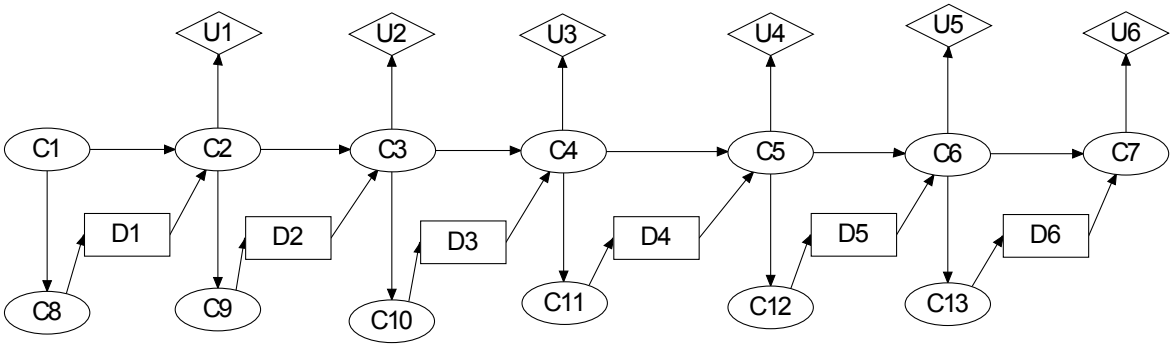


Figure 3: A POMDP.

by approximating δ_6 through enhancing C_6 (see Figure 4), and the largest policy domain when solving the ID will contain four variables (δ_5 has the domain $\{D_3, C_{11}, D_4, C_{12}\}$) .

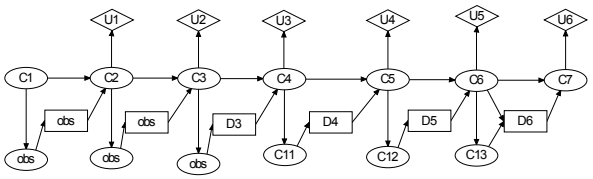


Figure 4: C_6 is enhanced for D_6. With the past of D_3 instantiated, the largest policy domain contains four variables.

You may also choose to approximate δ_5 through enhancing C_5 (Figure 5), and the largest policy domain when solving the ID contains three variables ($\delta_6(C_5, D_5, C_{13})$).

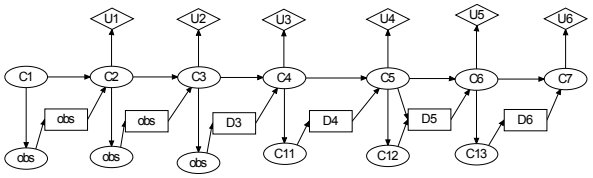


Figure 5: C_5 is enhanced for D_5. Now, the largest domain contains three variables.

As a small test of the approach we tried the three structures above (with only binary variables) with three different arbitrary parameter settings, and with utilities after each move as well as with utility after the last move, only. We looked at the policy δ_3, and for all cases, the optimal policy was the same as the approximated one.

2.1 Maximize uncertainty

Consider the general situation as described in Figure 2. If we wish to approximate D_{10} by information enhancement, we can enhance the pair $(11, 13)$ as well as $(11, 14)$ - blocking for everything but 10. (See Figure 6). When discussing IDs we shall use the terms 'variable' and 'node' interchangeably.

The node 14 is further away from the utility node than 13, and therefore, disclosing 13 will give you more certainty of the expected utility than would disclosing 14. This means that adding the information 14's state brings you closer to the actual knowledge at the time of deciding D_{10} than would adding the information of 13's state, and enhancing $(11, 14)$ is a better approximation than enhancing $(11, 13)$.

We have performed a small experiment with the ID in Figure 2 and approximated δ_7 with the optimal policy from Figure 6. All nodes were binary. Out of the 64 configurations of the domain, the policies coincided on 61 cases. For one case the approximated policy has a tie between the correct decision and another one, in the two other cases, the difference in EU between the correct and the approximated decision was 0.001 on a value around 50 (on a scale from 0 to 100).

3 Border and Frontier

In general, we have two decision nodes D_i and D_j $(i < j)$ in an influence diagram. The set of disclosed variables at the time of deciding D_i is denoted $\mathcal{P}$. We should index the set with i, but for notational convenience we will skip the indices i and j. The set of nodes becoming disclosed between deciding D_i and D_j (including D_i) is denoted $\mathcal{I}nf$. $\mathcal{P}$ and $\mathcal{I}nf$ are the disclosed nodes. With $i = 7$ and $j = 10$ in Figure 2 we have that $\mathcal{P}$ is the nodes P_1 to P_6 and $\mathcal{I}nf = \{D_7, D_8, D_9, 8, 10, 16, 21, 22, 23\}$. The set of descendants of D_j is denoted $\mathcal{D}$. Only the utility nodes in $\mathcal{D}$ are relevant for D_j. They are denoted $\mathcal{U}$. In Figure 2, $\mathcal{D} = \{12, 24, U\}$ and $\mathcal{U} = \{U\}$.

The scene is now that the utility nodes of interest are $\mathcal{U}$, and we look for closed nodes, which if disclosed would turn some nodes in $\mathcal{I}nf$ irrelevant. That is, we search for cut sets $\mathcal{C}$ such that $\mathcal{U}$ is d-separated from $\mathcal{I}nf$ given $\mathcal{C}$ (we define d-separation such that nodes from $\mathcal{I}nf$ are allowed in $\mathcal{S}$). The chance nodes in $\mathcal{D}$ can not be used in such cut sets as this would create a directed cycle.

The basic idea is to establish two cut sets, the *border* and the *frontier*. The border is the smallest cut set of non-$\mathcal{D}$ chance nodes closest to $\mathcal{U}$.

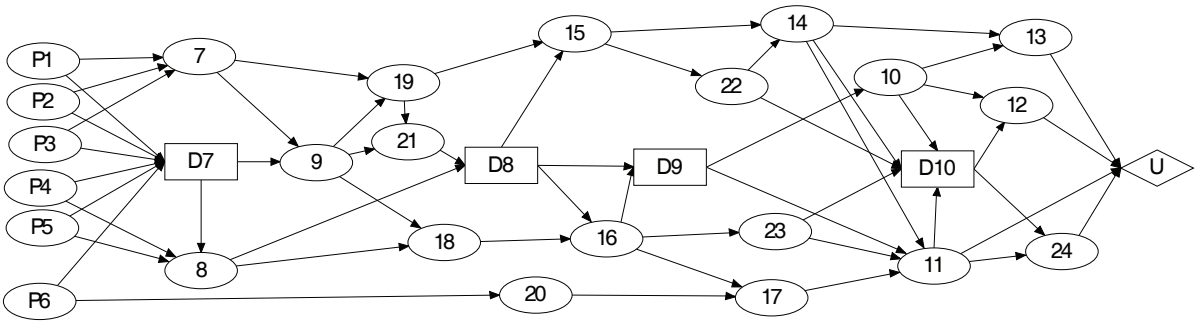


Figure 6: The ID in Figure 2 with the nodes 11 and 14 enhanced.

Definition 1. A node $X \notin \mathcal{D}$ *belongs* to the border if

- X is a parent of an element of $\mathcal{D}$
- There is an active path from $\mathcal{I}nf$ to X

The set of border nodes is denoted by $\mathcal{B}$.

In Figure 2 the border consists of the nodes $\{10, 11, 13\}$.

As none of the descendants of $\mathcal{B}$ are disclosed before deciding D_j we have:

Proposition 1. *$\mathcal{I}nf$ is d-separated from $\mathcal{U}$ given $\mathcal{B}$*

Knowing that $\mathcal{B}$ is a cut set, you may go backwards from $\mathcal{B}$ in the network to create new cut sets. Actually, all chance nodes on active paths from $\mathcal{I}nf$ to $\mathcal{B}$ may be part of a cut set. The task is to identify the relevant part of the network and for this relevant part to identify good cut sets for information enhancement.

Definition 2. A network is *regular* if there is no active path from $\mathcal{I}nf$ to $\mathcal{U}$ involving a converging connection over a node in $\mathcal{P}$ or with a descendant in $\mathcal{P}$.

The network in Figure 2 is regular, and in the following sections we assume the network in consideration to be regular.

Definition 3. The set of *information holders*, $\mathcal{I}$, consists of all closed nodes with a directed path of closed nodes to $\mathcal{I}nf$.

For the network in Figure 2 we have $\mathcal{I}$= $\mathcal{I}nf \cup \{7, 8, 9, 15, 18, 19\}$.

Definition 4. A node in $\mathcal{I}$ that has a directed path of closed nodes to $\mathcal{U}$ and with no intermediate nodes in $\mathcal{I}$ is said to belong to the *frontier* of D_j. The set of frontier nodes is denoted by $\mathcal{F}$.

In Figure 2 the frontier of D_{10} consists of the nodes $\{10, 15, 16, 22, 23, D_9\}$.

Theorem 1. *$\mathcal{I}$ is d-separated from $\mathcal{U}$ given $\mathcal{F}$.*

Proof. Let $V_0 \in \mathcal{I}$, $U \in \mathcal{U}$, and let $\langle V_0, \ldots, V_k, U \rangle$ be an active path given $\mathcal{F}$.

Assume that $\langle V_0, \ldots, V_k, U \rangle$ contains a converging connection, and let $V_{s-1} \rightarrow V_s \leftarrow V_{s+1}$ be the last converging connection on the path from V_0 to U. As $V_s \notin \mathcal{P}$ nor has a descendant in $\mathcal{P}$, V_s or one of its descendants is disclosed, and hence $V_s \in \mathcal{I}$. Therefore, also $V_{s+1} \in \mathcal{I}$. When you follow the path towards U you will meet a diverging connection $V_{t-1} \leftarrow V_t \rightarrow V_{t+1}$. Then $V_t \in \mathcal{F}$, and the path is not active. Note that you will meet a diverging connection at the latest when you reach $V_k \rightarrow U$. We conclude that there is no converging connections on the path.

Assume that the first link is $V_0 \leftarrow V_1$. Then, follow the path until you reach a diverging connection. As there are no converging connections on the path, there must be exactly one diverging connection $V_{s-1} \leftarrow V_s \rightarrow X$. Then $V_s \in \mathcal{F}$ and the path is not active.

To conclude: the active path is directed from V_0 to U, and it cannot contain intermediate nodes from $\mathcal{F}$. Therefore $V_0 \in \mathcal{F}$.

□

From the proof above we can conclude that information from $\mathcal{I}nf$ flows to U through a path against the direction of the links followed by a path along the links. The node, where the direction of the flow turns, is a frontier node.

3.1 Finding the border and the frontier

There are two obvious candidate sets for enhancement, namely $\mathcal{B}$ and $\mathcal{F}$. They are determined through a sequence of graph searches (for example breath-first search). First you determine $\mathcal{D}$ and $\mathcal{U}$, by starting a breath first search from the decision D_j. All chance nodes reached are labeled D. They are the elements of $\mathcal{D}$, and the utility nodes are the elements of $\mathcal{U}$. The nodes in $\mathcal{D}$ cannot be enhanced as this will introduce a directed cycle. The non-$\mathcal{D}$ parents of the nodes in $\mathcal{D} \cup \mathcal{U}$ are the candidate border nodes, and they are labeled CB.

Next, start a backwards breath-first search from each of the decision nodes $D_{i+1}, \ldots, D_j$. That is, you follow the edges opposite to their direction. You stop when you meet a previous decision node or a node in $\mathcal{P}$. Each node you meet is labeled with an I. Perform a backwards breath-first search from the nodes of $\mathcal{D}$. When you meet a node X with label I you give it the label F, and break the search behind X. Finally, perform a breath-first search from $\mathcal{F}$. When you meet a node X with label CB, change the label to B and stop searching behind X. The various labels for the ID in Figure 2 are given in Figure 7.

4 Cut sets between frontier and border

There may be other candidate sets for enhancement than $\mathcal{B}$ and $\mathcal{F}$. Actually, any set of nodes which d-separates the frontier from the border can be used for enhancement.

Proposition 2. *Let $\langle V_0, \ldots, V_k \rangle$ be an active path with $V_0 \in \mathcal{F}$ and $V_k \in \mathcal{B}$. Then the intermediate nodes cannot be in $\mathcal{D}$ nor in $\mathcal{I}$, and the path is directed from V_0 to V_k.*

Proof. The proof of Theorem 1. □

Definition 5 (Free graph)**.** The subgraph $\mathcal{G}$ consisting of $\mathcal{F}$, $\mathcal{B}$ and all nodes on a directed path from a node in $\mathcal{F}$ to a node in $\mathcal{B}$ is called the *free graph* (see Figure 8) .

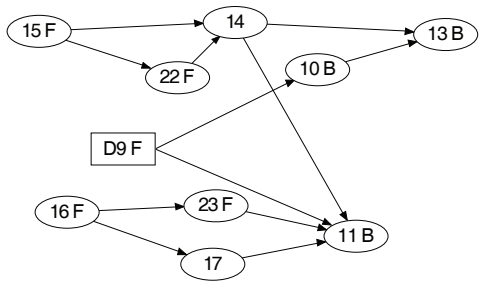


Figure 8: The free graph for the ID in Figure 2.

The proposition yields that we can use any set in $\mathcal{G}$ which d-separates $\mathcal{F}$ from $\mathcal{B}$. Unfortunately, finding all possible cut sets may for a large $\mathcal{G}$ be intractable. If that is the case, the following heuristics can in polynomial time provide a set of very good candidates for information enhancement.

Note that you cannot just perform a flow analysis on the directed graph $\mathcal{G}$. In Figure 8, for example, the set $\{10, 11, 15, 22\}$ does not block for the information coming from 16 or 23.

4.1 Cut set heuristics

To indicate that the information is flowing to $\mathcal{B}$, you extend the free graph with dummy children U_i of the nodes in $\mathcal{B}$. See Figure 9.

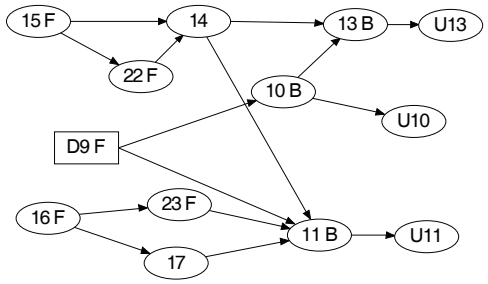


Figure 9: The extended free graph for the ID in Figure 2.

Next, triangulate the extended free graph and form a junction tree.

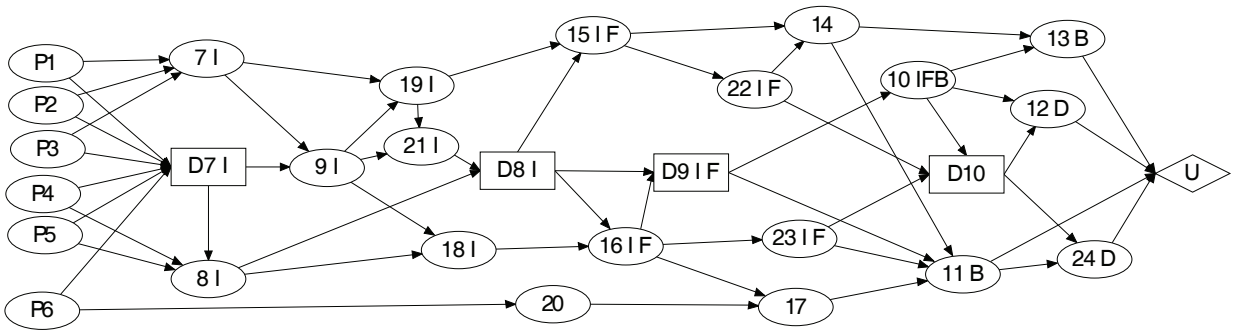


Figure 7: The ID marked after the search algorithms. "D" indicates that the node cannot be enhanced; "I" indicates nodes with information to transmit; "B" indicates border; "F" indicates frontier.

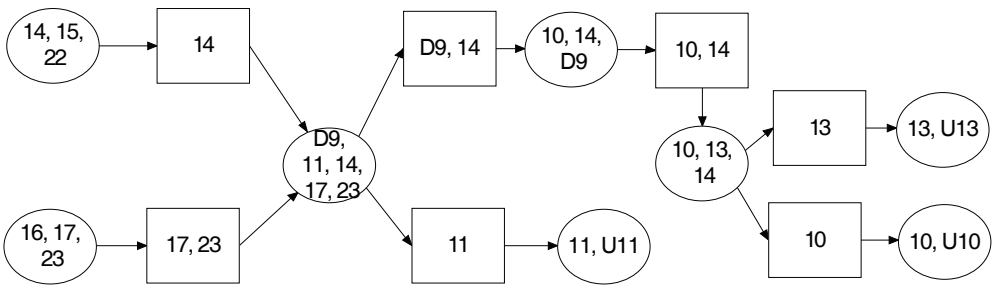


Figure 10: A junction tree for the extended free graph in Figure 8.

The junction tree will provide separators, which can be used to determine cut sets. You look for sets of separators blocking the flow from frontier nodes to border nodes. You start in the leafs with U-nodes and move backwards.

Consider the junction tree in Figure 10. The separators 10, 11, and 13 d-separate the U-nodes from the rest. They form $\mathcal{B}$. As 10 is a frontier node, you cannot block it with nodes further back in the junction tree. Going backwards from the clique (10, 13, 14) you meet the separator (10, 14), and you find the cut set (10, 11, 14). Going backwards from the separator (10, 14), you meet a clique with the frontier node D_9, and therefore you hereafter have to include D_9 in the cut sets. The same happens when you go backwards from the separator 11. The the new cut sets $\{10, 11, 14\}$, and $\{10, 14, 17, 23, D_9\}$.

4.2 Irregular networks

If the network is irregular, the search for frontier nodes is more involved. An active path from $\mathcal{I}nf$ to $U \in \mathcal{U}$ ends with a directed series $V_k \rightarrow \ldots \rightarrow U$. The first node in this series is a frontier node. For regular networks, the frontier consists of common ancestors of $\mathcal{B}$ and $\mathcal{I}nf$. For irregular networks we need to define $\mathcal{I}$ differently: for $X \rightarrow Y \leftarrow Z$ with $Y \in \mathcal{I}$ and with $Z \in \mathcal{P}$ or with a descendant in $\mathcal{P}$ we also include X in $\mathcal{I}$.

4.3 An iterative procedure

You may choose the nodes in the cut set iteratively, and whenever a node has been selected, you may renew the analysis. In the example for this paper it is certain that node 10 always will be part of the domain for δ_{10}. Hence, we need not look for ways of blocking information coming from 10, and the ancestors of 10 are only relevant if they are ancestors of other information nodes. Actually, for the ID in Figure 2, inclusion of 10 does not change the analysis because the only parent of 10 is D_9, and it is an information node.

5 Conclusions and future work

We have established methods for finding approximate representations of future decisions policies through information enhancement. The methods do not determine all possible candidates for information enhancement. First of all, a good cut set does not necessarily contain only nodes between the border and the frontier. That is, you may go behind the frontier.

Furthermore,we have only treated approximation of the last decision. Usually, the decision in question has several future decisions to consider, and you may look for a combined approximation of several future decisions.

Acknowledgments

Thanks to the anonymous referees for constructive feedback and to the Machine Intelligence Group at Aalborg University for fruitful discussions - thanks in particular to Thorsten Ottosen.

References

A. W. Drake (1962). Observation of a Markov process through a noisy channel. *Ph. D. thesis, MIT, Dept of Electrical Engineering*

Finn V. Jensen (2008). Approximate representation of optimal strategies from influence diagrams *Proceedings of the 4th European Workshop on Probabilistic Graphical Models.*.

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.

Dennis Nilsson and Steffen L. Lauritzen (2001). Representing and solving decision problems with limited information. *Management Science*, 47: 1235-1251.

S. M. Olmsted (1983). On Representing and Solving Decision Problems *Ph.D. thesis, Department of Engineering-Economic Systems, Stanford University.*

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.

Robust Independence-Based Causal Structure Learning in Absence of Adjacency Faithfulness

Jan Lemeire Stijn Meganck Francesco Cartella
ETRO Department, Vrije Universiteit Brussel, Belgium
Interdisciplinary Institute for Broadband Technology (IBBT), Belgium
{*jan.lemeire, stijn.meganck, francesco.cartella*}*@vub.ac.be*

Abstract

This paper presents an extension to the Conservative PC algorithm which is able to detect violations of adjacency faithfulness under causal sufficiency and triangle faithfulness. Violations can be characterized by pseudo-independent relations and equivalent edges, both generating a pattern of conditional independencies that cannot be modeled faithfully. Both cases lead to uncertainty about specific parts of the skeleton of the causal graph. This is modeled by an f-pattern. We proved that our Very Conservative PC algorithm is able to correctly learn the f-pattern. We argue that the solution also applies for the finite sample case if we accept that only strong edges can be identified. Experiments based on simulations show that the rate of false edge removals is significantly reduced, at the expense of uncertainty on the skeleton and a higher sensitivity for accidental correlations.

1 Introduction

Independence-based algorithms for learning the causal structure from data rely on the Conditional Independencies (CIs) entailed by the system's causal structure. The *causal Markov condition* gives the CIs that follow from a causal structure that is represented by a Directed Acyclic Graph (DAG): every variable is independent of its non-effects conditional on its direct causes. All algorithms rely on a form of faithfulness. *Causal faithfulness* says that no other CIs appear in the system's probability distribution than those entailed by the causal Markov condition. Faithfulness is therefore a very convenient property: all CIs tell us something about the causal structure. Violation of faithfulness means that there are non-Markovian CIs.

The validity of causal faithfulness is supported by the 'Lebesgue measure zero argument' (Meek, 1995b), which says that the chance of randomly picking a parameterization of a Bayesian network resulting in non-Markovian CIs has measure zero. But in near-to-unfaithful situations probability distributions come infinitely close to unfaithful distributions, such that a test for independence which has to rely on a finite sample will not be able to identify the dependencies correctly. The Lebesgue measure zero argument does not hold here, since the ϵ-regions around unfaithful situations do not have Lebesgue measure zero.

(Zhang and Spirtes, 2007) showed that only in cases of *triangle unfaithfulness* violations of faithfulness are *undetectable*. This happens when the true probability distribution is not faithful to the true causal DAG, but is nonetheless faithful to some other DAG. In those cases, the CIs do not give enough evidence to learn the correct DAG. This will be discussed in more detail in the next section. We will therefore have to assume triangle faithfulness. Then, violations of faithfulness are detectable in the sense the true probability distribution is not faithful to any DAG. It means that there exist several DAGs that each explain a subset of the CIs.

(Ramsey et al., 2006) showed that we only need adjacency faithfulness and orientation faithfulness to learn the correct equivalence class. *Adjacency Faithfulness* states that any two adjacent variables do not become independent when conditioned on some other (possible empty) set of variables. It is necessary to recover the correct skeleton of the true DAG. *Orientation faithfulness* (check reference for definition) is necessary for finding the correct orientations. (Ramsey et al., 2006) extended the well-known PC algorithm to detect violations of orientation faithfulness. Violations lead to specific ambiguous parts of the DAG, in which no decision on the orientation can be taken. The Conservative PC algorithm is given in the next section. In this paper we apply the same idea for handling violations of adjacency faithfulness. They can be identified, under triangle faithfulness, by two patterns: *pseudo-independent relations* and *equivalent edges*. These patterns will lead to parts of the model in which no decision can be taken on the correct skeleton.

The following section recalls the important aspects of independence-based causal structure learning. In section 3 we analyze violations of adjacency faithfulness. Based on the identified CI patterns, the VCPC algorithm is presented and proven to be correct in section 4. Section 5

Pp. 169–177 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

analyzes the finite sample case. Finally, the experimental results are presented in section 6.

2 Independence-Based Causal Inference

We recall the Conservative PC algorithm (CPC), see Alg. 1. $Adj(G, X)$ denotes the set of nodes adjacent to X in graph G. Single stochastic variables are denoted by capital letters, sets of variables by boldface capital letters. Step 3 consists of extensions to the original PC algorithm (Spirtes et al., 1993) in which Orientation-Faithfulness is tested (Ramsey et al., 2006). Edges of an unshielded triple, i.e. a triple $< X, Y, Z >$ for which X and Z are both adjacent to Y, but X and Z are not adjacent, are not oriented if a failure is detected, but are indicated as unfaithful, as shown in Fig. 1(a). An e-pattern is a partially-oriented DAG in which some triples are denoted as unfaithful.

Undetectable violations of faithfulness only happen by violations of the triangle faithfulness (Zhang and Spirtes, 2007) condition. It states that given a set of variables V whose true causal DAG is G, let X, Y, Z be any three variables that form a triangle in G

1. If Y is a non-collider on the path $< X, Y, Z >$, then X, Z are dependent conditional on any subset of $V \backslash \{X, Z\}$ that does not include Y.
2. If Y is a collider on the path $< X, Y, Z >$, then X, Z are dependent conditional on any subset of $V \backslash \{X, Z\}$ that includes Y

To illustrate triangle unfaithfulness, consider the DAG shown in Figure 2(b). There are 3 ways to violate triangle faithfulness for this DAG:

(TRUFF1) $X \perp\!\!\!\perp Y$ gives faithful model $X \rightarrow Z \leftarrow Y$

(TRUFF2) $Y \perp\!\!\!\perp Z$ gives faithful model $Y \rightarrow X \leftarrow Z$

(TRUFF3) $X \perp\!\!\!\perp Z | Y$ gives faithful model $X \rightarrow Y \rightarrow Z$

Besides faithfulness, *minimality* (MIN) is also a basic condition: elimination of an edge leads to a Bayesian network which violates the Markov condition. Formally:

$$\forall X, Y \in \boldsymbol{V} \text{which are adjacent in Bayesian network :} \quad X \not\perp\!\!\!\perp Y \mid OthPa(X–Y) \tag{1}$$

where $OthPa(X-Y)$ of edge $X-Y$ is defined as $Parents(Y) \setminus X$ if X is parent of Y, otherwise it is $Parents(X) \setminus Y$. $OthPa$ is short for 'other parents'.

3 Violation of Adjacency Faithfulness

Here we analyze unfaithfulness in the case of a perfect test for (conditional) independence. Later we will consider imperfect tests due to finite sample sizes. In all other cases than triangle unfaithfulness, there are Conditional Independencies (CIs) that make violation of faithfulness detectable. Violations of adjacency faithfulness can be identified by two patterns: pseudo-independent relations and information equivalences. Consider $X \rightarrow Y$. There are 2 kinds of violations: one in which X and Y are marginally independent and one in which they become independent when conditioned on some $\boldsymbol{Z}$.

Algorithm 1 The CPC algorithm

S1 Start with the complete undirected graph U on the set of variables $\boldsymbol{V}$.

Part I ***Adjacency search***.

S2 $n = 0$;

repeat

For each pair of variables A and B that are adjacent in (the current) U, check through the subsets of $Adj(U, A) \setminus \{B\}$ and the subsets of $Adj(U, B) \setminus \{A\}$ that have exactly n variables. For all such subsets $\boldsymbol{S}$ check independency $A \perp\!\!\!\perp B \mid \boldsymbol{S}$. If independent, remove the edge between A and B in U, and record $\boldsymbol{S}$ as $Sepset(A, B)$;

$n = n + 1$;

until for each ordered pair of adjacent variables A and B, $ADJ(U, A) \setminus \{B\}$ has less than n elements.

Part II ***Orientation.***

S3 Let G be the undirected graph resulting from step S2. For each unshielded triple $\langle A, B, C \rangle$ in G, check all subsets of A's potential parents (nodes that are adjacent to A but are not A's children) and of C's potential partners:

(a) If B is NOT in any such set conditional on which A and C are independent, orient the triple as a collider: $A \rightarrow B \leftarrow C$;

(b) If B is in all such sets conditional on which A and C are independent, leave $A - B - C$ as it is , i.e., a non-collider;

(c) Otherwise, mark the triple as "unfaithful" by underlining the triple, $A \underline{-B-} C$.

S4 Execute the orientation rules given in (Meek, 1995a), but not on unfaithful triples.

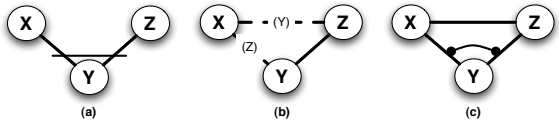


Figure 1: The three cases of uncertainty: (a) an unfaithful triple by violation of orientation faithfulness for unshielded triple $\langle A, B, C \rangle$, (b) PPIRs when $X \perp\!\!\!\perp Y$ in model $X \rightarrow Y \leftarrow Z$ and (c) equivalent edges when $X \perp\!\!\!\perp Y | Z$ in model $X \rightarrow Y \rightarrow Z$.

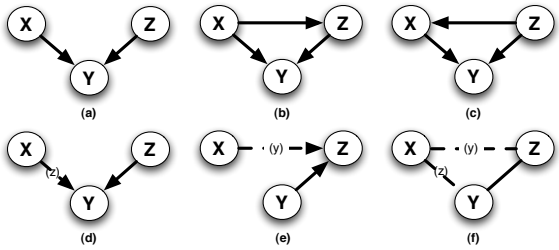


Figure 2: Marginal independency $X \perp\!\!\!\perp Y$ leads to violation of triangle faithfulness for (b) and (c). For (a) this gives a PIR, denoted in (d). This model is equivalent to (e). Both equivalent structures are denoted by PPIRs in (f).

3.1 Violation by Marginal Independence

Whenever $X \perp\!\!\!\perp Y$ for some adjacent variables X and Y, we call $X - Y$ a *Pseudo-Independent Relation* (PIR). Then, by Eq. 1, there exists at least one subset of $\boldsymbol{V}$, namely $OthPa(X - Y)$, which turns the independency into a dependency after conditioning. We call any such subset a dependency set, or *depset* for short, of X and Y, written as $depset_{XY}$. A special case in which PIRs occur is identified as *pseudo-independent models* (Xiang et al., 1996), in which three variables are pairwise independent but become dependent when conditioned on the third variable.

For not overloading the rest of the discussion we assume that for each PIR there exists a depset with one element. PIRs with larger depsets can be identified similarly, but such cases are very rare.

Assumption 1 *If X and Y are adjacent, and $X \perp\!\!\!\perp Y$, there exists a $Z \in OthPa(X–Y)$ such that $X \not\perp\!\!\!\perp Y | Z$.*

Take $Z \in OthPa(X - Y)$ which forms a depset of $X–Y$. Z is adjacent to Y. If Z would also be adjacent to X, $X \perp\!\!\!\perp Y$ is a result of triangle unfaithfulness (TRUFF1 or TRUFF2), as shown in Fig. 2 by (b) and (c). By excluding the triangle case, $X \rightarrow Y \leftarrow Z$ is an unshielded collider for which there is a $\boldsymbol{U}$ such that $X \perp\!\!\!\perp Z | \boldsymbol{U}$ and in general $X \not\perp\!\!\!\perp Z | Y, \boldsymbol{U}$. Fig. 2(a) shows such a model for which $\boldsymbol{U}$ is empty. To denote a PIR, we annotate the edge with the depset, as shown in Fig. 2(d).

A PIR implies a marginal independence and a conditional dependence. This pattern is the same as that of a v-structure. Hence, a PIR leads to two equivalent structures that can explain all CIs. Fig. 2(e) gives the same CIs as (d). We describe this pattern by connecting variables which are marginally independent but have a depset by a special edge: a Potential PIR (PPIR). A PPIR is written as $X–(Z)–Y$ and graphically denoted by a dashed edge annotated with the depset, as shown in Fig. 2(f). A PPIR can thus be a PIR or be part of a v-structure. In Sec. 4 we will see that in certain cases, a PIR can be identified from a structure with 2 PPIRs.

3.2 Violation by a non-Markovian Conditional Independence

A second violation of adjacency faithfulness happens when for adjacent variables X and Y: $X \not\perp\!\!\!\perp Y | depset_{XY}$ and there is a set $\boldsymbol{Z}$ for which $[X \perp\!\!\!\perp Y \mid \boldsymbol{Z} \cup depset_{XY}]$. The latter denotes a *strict CI*: a CI that turns into a conditional dependency for each proper subset of the conditioning set. Based on this independence, the PC algorithm would wrongly remove the edge between X and Y. We will prove that under triangle faithfulness (1) there are CIs that let us detect such false separations, and (2) the ambiguities can be represented by *equivalent edges*. We first define equivalent edges and present an example. For clarity, we omit the depset $depset_{XY}$ in the discussion.

3.2.1 Equivalent edges

The result of a false separation given by the above strict CI are 2 or more equivalent structures in which one edge can be replaced by another. We call them *equivalent edges*. Equivalent edges are linked with an arc with a bullet at each end, as shown in Fig. 1(c).

Definition 2 *Take distribution P and G a DAG not containing directed edges X–Y and Z–Y. Two edges X–Y and Z–Y are called equivalent edges if and only if G is not Markovian for P and*

$$\begin{aligned} & G \cup X–Y \text{ is Markovian for } P \\ \Leftrightarrow\ & G \cup Z–Y \text{ is Markovian for } P \end{aligned} \tag{2}$$

Note that with $X–Y$ we denote that the edge can have both orientations. A DAG is called Markovian for a distribution if all CIs of the DAG given by the Markov condition are present in the distribution.

3.2.2 Example of Information Equivalence.

Consider the structure $Z \rightarrow X \rightarrow Y$ and the deterministic relation $X = f(Z)$. Two conditional independencies follow:

$$Z \perp\!\!\!\perp Y \mid X \ \ \& \ \ X \perp\!\!\!\perp Y \mid Z. \tag{3}$$

We call Y and Z *information equivalent* with respect to X (Lemeire, 2007). Since $X \not\perp\!\!\!\perp Y$, this a violation of the *intersection condition* (Pearl, 1988). The first equation comes from the Markov condition, the second is implied by the functional relation. X is completely determined by Z, so Z has all information about X. Knowing Z therefore renders X irrelevant for Y. Information equivalences happen when there are deterministic relations, but also under weaker conditions (Lemeire, 2007).

In the example, we have $[X \perp\!\!\!\perp Y \mid Z]$ which falsely suggests that Z separates X from Y and edge $X \rightarrow Y$ can be removed. But $Z \perp\!\!\!\perp Y \mid X$ suggests that $Y - Z$ can be removed. Removal of both edges results in a non-Markovian DAG. An ambiguity on the correct structure is a result. X or Z should be connected to Y to explain the dependencies. Structures $Z \rightarrow X \rightarrow Y$ and $X \leftarrow Z \rightarrow Y$ are equivalent given the CIs. $X - Y$ and $Z - Y$ are equivalent edges.

Concluding, CI $Z \perp\!\!\!\perp Y \mid X$ made it possible to identify a strict CI that would lead to a false separation. In the following section we present the general conditions and prove that they lead to equivalent edges.

3.2.3 Conditions for equivalent edges

When strict CI $[X \perp\!\!\!\perp Y|\mathbf{Z}]$ is observed, removal of $X - Y$ is only valid when $\mathbf{Z}$ is a *minimal cut set*[1] in the true graph. The following theorem gives the conditions to recognize a 'false minimal cut set'. A strict d-separation, denoted as $[X \perp Y|\mathbf{Z}]$, is a d-separation which gives a d-connection for any proper subset of $\mathbf{Z}$.

Theorem 3 *$\mathbf{Z}$ is not a minimal cut set for X and Y in G if for one of the elements U of $\mathbf{Z}$ one of the following d-separations hold in G: ($\mathbf{Z'} = \mathbf{Z} \setminus U$ and $\boldsymbol{T} \subset V \setminus \mathbf{Z} \setminus \{X, Y\}$)*

1. *$U \perp Y|\mathbf{Z'}$ or $U \perp X|\mathbf{Z'}$;*
2. *$U \perp Y|\mathbf{Z'}, \boldsymbol{T}$ and $U \perp X|\mathbf{Z'}, \boldsymbol{T}$;*
3. *$[U \perp Y|X, \mathbf{Z''}]$ or $[U \perp X|Y, \mathbf{Z''}]$ for some $\mathbf{Z''} \subset \mathbf{Z'}$;*
4. *$[U \perp Y|X, \mathbf{Z'}, \boldsymbol{T}]$ or $[U \perp X|Y, \mathbf{Z'}, \boldsymbol{T}]$.*

If none of the d-separations hold, either $\mathbf{Z}$ is a subset of a minimal cut set, or there is a $U \in \mathbf{Z}$ that forms a triangle or a v-structure with X and Y.

Proof:
To be a minimal cut set, all elements of $\mathbf{Z}$ must lie on a separate path between X and Y, and all paths between X and Y must be blocked by $\mathbf{Z}$. With path we mean an active path in terms of a d-connection. The conditions happen when $\mathbf{Z}$ is not a minimal cut set. Condition (1) or (2) hold when U is not connected to X or Y with a separate path. Condition (3) or (4) happen when U is d-connected to Y via X (by the strictness).

The last part is about what happens when none of the conditions are met. Conditions (1) or (2) guarantee that all elements of $\mathbf{Z}$ are connected with X and Y via separate paths. Next for a cut set, all paths between X and Y must be cut. If there would be an uncut path via another node, this node should be added to $\mathbf{Z}$ to form a cut set. The remaining case is when X and Y are adjacent. Then, unless U forms a triangle with X and Y, there exists a subset $\boldsymbol{T}$ which separates U from Y (or X). U can then be d-separated from Y given X, $\boldsymbol{T}$ and $\mathbf{Z'}$ (condition (4)) unless they form a v-structure ($U \rightarrow X \leftarrow Y$). ∎

The CIs corresponding to the d-separations of the theorem can be used to detect false separations. Condition (1) will be used in the finite sample case discussed in Section 5. The CIs corresponding to conditions (3) and (4) result in the presence of equivalent edges, as shown by the following theorem.

Theorem 4 *If G containing edges X–Y and U–Y is a Markovian DAG for P, $[X \perp\!\!\!\perp Y|\mathbf{Z'} \cup U]$ and one of the CIs corresponding to conditions (3) and (4) holds for U, then X–Y and U–Y are equivalent edges.*

Proof:
First we prove that $G \setminus X$–Y is a Markovian DAG. To prove this, assume $A \not\perp B|\boldsymbol{S}$ which holds in G but would not be represented in $G \setminus X$–Y. For this, the only active path from A to B must go via X–Y and no path may exist via $U - Y$. It follows that $A \perp B|\boldsymbol{S} \cup X$ (a). A is related to X but cannot be related to U since otherwise there would be a path to Y via U–Y. This could only happen if U is a collider on the path between A and Y. We prove that this results in a contradiction. From $Y \rightarrow U$ follows that the path from U goes towards X to have an active path from Y to X through U (to represent the dependencies given by the strict CI). Acyclicity gives then $Y \rightarrow X$. The active path from X to A must then be pointing towards A for having an active path between A and B. This, however, creates a path from Y to A via U and X, which was excluded. Hence $A \perp U|\boldsymbol{S}$ (b). From the given CI and the CIs following from (a) and (b) follows that $A \perp\!\!\!\perp B|\boldsymbol{S}$ which results in a contradiction.

Next, by conditions (3) or (4), U could be d-separated from Y by X. But if we would also remove U–Y, the dependency $X \not\perp Y|\mathbf{Z'}$ is not present anymore, since the path from X to Y via U is removed. The graph without both edges is thus not Markovian. The graph $G \setminus U$–Y is also Markovian by swapping X and U in the proof. ∎

The simplest case of Condition (3), with an empty $\mathbf{Z''}$, was discussed in Section 3.2.2. Fig. 3 gives an exam-

[1] A cutset is a set of variables which blocks all active paths between X and Y. A cutset is minimal if no proper subset is a cutset.

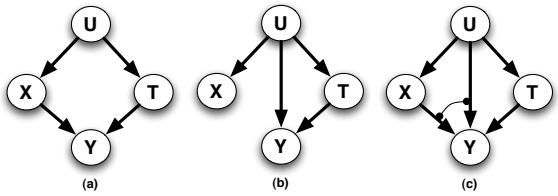


Figure 3: Example of two equivalent structures. If $X \perp\!\!\!\perp Y|U$ holds for (a) and $U \perp\!\!\!\perp Y|X,T$ for (b), than both represent the same CIs. $X-Y$ and $U-Y$ are equivalent edges, which is denoted in (c).

ple of Condition (4) with a non-empty $\boldsymbol{T}$. Assume that the non-Markovian CI $[X \perp\!\!\!\perp Y|U]$ holds. In that case, the CPC algorithm will delete edge $X-Y$. Since at that point of the algorithm Y is still connected to U, all observed dependencies are explained. But with Markovian independency $U \perp\!\!\!\perp Y|X \cup T$, deletion of edge $Y-U$ results in a model which cannot explain the dependencies. We end up with two equivalent structures: one with edge $X-Y$, the other with edge $Y-U$. Both models explain all dependencies. But the first cannot explain $X \perp\!\!\!\perp Y \mid Z$, the second cannot explain $Z \perp\!\!\!\perp Y|X \cup U$.

To simplify the rest of the discussion we will exclude the equivalences following from a condition with a non-empty ***Z"*** in condition (3) of the theorem. They can be treated in a similar way, but are much rarer.

Assumption 5 *For all strict independencies of the form* $[X \perp\!\!\!\perp Y|U \cup \boldsymbol{Z} \cup depset_{XY}]$ *(with* $\boldsymbol{Z'} \subset \boldsymbol{Z}$*):*

$$Y \perp\!\!\!\perp U|X \cup \boldsymbol{Z'} \cup depset_{XY} \Rightarrow Y \perp\!\!\!\perp U|X \cup depset_{XY}.$$

3.2.4 Relation to NPC

The necessary path condition (NPC) algorithm (Steck and Tresp, 1999) was introduced as a robust extension for the PC algorithm. It states that for each strict conditional independence $[X \perp\!\!\!\perp Y|\boldsymbol{Z}]$ there must exist a path between X (Y) and each $U \in \boldsymbol{Z}$ not crossing Y (X). This is similar to the notion described above that the minimal cutset of two variables needs to be connected to both variables. The NPC introduces the concept of ambiguous edges, which is defined as an edge whose presence depends on the absence of another. In NPC, these ambiguous regions are resolved by including a minimal number of ambiguous edges in order to satisfy a maximal number of independence relations. In our case, ambiguous regions correspond to the equivalences we find between edges. Instead of forcing them into a DAG structure, we model the ambiguity explicitly by an f-pattern. An *f-pattern* is an e-pattern augmented by edges that are denoted as PPIRs and subsets of edges denoted as equivalent. An oriented PPIR in the pattern is identified as a PIR.

3.3 Augmented Knowledge Graph

We will use an augmented knowledge graph to model the causal information. We define an *augmented* knowledge graph (AKG) (Eberhardt, 2008) as a graph containing the following relations between any two variables X and Y: $X \;\; Y$, $X \rightarrow Y$, $X-Y$, $X-(\boldsymbol{S})-Y$ and $X-(\boldsymbol{S}) \rightarrow Y$, with $\boldsymbol{S}$ a set of sets of variables. Furthermore, edges can be also related with one another either by a straight line ——— or a curved line with round endpoints •—•.

An f-pattern can be represented using an augmented knowledge graph by using the following interpretations for the different relations between variables

$X \;\; Y$ Neither X nor Y are direct causes of one another.

$X \rightarrow Y$ X is a direct cause of Y.

$X - Y$ Either X is a direct cause of Y or reverse.

$X-(\boldsymbol{S}) \rightarrow Y$ There is a pseudo-independent causal relationship between X and Y with $\forall D \in \boldsymbol{S}$, D is a depset for the PIR.

$X-(\boldsymbol{S})-Y$ There is either a pseudo-independent, direct causal or no relation between X and Y

and the following interpretations for the relation between edges:

X —Y— Z The triple $< X, Y, Z >$ is unfaithful.

X —Y— Z The edges $X-Y$ and $Z-Y$ are equivalent.

4 The Very Conservative PC Algorithm

The Very Conservative PC algorithm (VCPC) adds to CPC the rules of S2' to S2 (Alg. 2) and replaces Part II with Part II' (Alg. 3). Besides recording the sepsets for pairs of variables, it will also record depsets. The algorithm returns an f-pattern. When we speak about adjacencies, these special edges are also considered. Non-equivalent and non-PPIR edges are called normal edges.

Theorem 6 *(Correctness of VCPC) Consider a graph G, a JPD P generated by G and $I(P)$, the set of CIs of P: if minimality, triangle-faithfulness and assumptions 1 and 5 hold for P, the algorithm will, based on $I(P)$, return an f-pattern describing a set of DAGs that includes G. The algorithm is not trivial; it does not always return the set of all DAGs.*

Proof:

1) Assume adjacency faithfulness:

Given adjacency faithfulness, the only difference with CPC during the adjacency search is that in step S2'[II]a for each true v-structure $X \rightarrow Z \leftarrow Y$, such that $X \perp\!\!\!\perp Y$, a PPIR $X-(Z)-Y$ is added. During the first part of the orientation phase these PPIRS are temporarily removed

Algorithm 2 VCPC algorithm S2'

[I] Before testing whether $X \perp\!\!\!\perp Y | \boldsymbol{S}$ holds, check the following:

a When $X - Y$ is a PPIR, add $depset_{XY}$ to $\boldsymbol{S}$.

b If $X - Y$ has an equivalent edge $X - Z$ or $Y - Z$ and Z is a member of $\boldsymbol{S}$, skip the test.

[II] If the independence test returns $X \perp\!\!\!\perp Y | \boldsymbol{S}$, do the following before removing the edge:

a If $\boldsymbol{S}$ is empty, look for a T in $Adj(X) \cup Adj(Y)$ for which $X \not\perp\!\!\!\perp Y \mid T$. If such a T exists, do not remove the edge, denote it as a PPIR with depset T.

b If $\boldsymbol{S}$ is not empty, test for all $Z \in \boldsymbol{S}$ whether $X \perp\!\!\!\perp Z | Y \cup depset_{XY}$ and $Z \perp\!\!\!\perp Y | X \cup depset_{XY}$. If for a Z, one of both independencies hold, do not remove edge $X - Y$ and do the following. Assume the first independency is found (if the second independency holds, just swaps X and Y in the following.). (1) If $X - Z$ has been removed due to d-separation by respectively Y or X, use this edge for constructing all sets $\boldsymbol{S}$ in S2 and add the edge back to the graph and go to (3). (2) If $X - Z$ has been removed due to some other d-separation, leave edge $X - Y$ in the graph, but do not qualify it as equivalent. (3) If $Y \not\perp\!\!\!\perp Z | depset_{XY}$, denote $X - Y$ as equivalent to $X - Z$ in the graph.

Algorithm 3 VCPC algorithm Part II'

Part II' ***Orientation.***

- Perform all of the following steps until no more edges can be oriented:

  Remove the PPIRs from G;

  Perform S3' as explained in (Ramsey et al., 2006), except that unshielded triples containing an equivalent edge are not considered;

  Perform S4 from the original algorithm on non-equivalent edges;

  Add the PPIR edges back G;

S5 Go through all PPIRs. Look for triangles consisting of normal edges and PPIRs in which for each PPIR the opposite variable in the triangle is a depset.

a If the triangle contains two normal edges which form a v-structure, remove the PPIR.

b If the triangle only contains one normal edge which is directed, direct the PPIR that contains the node to which the arrow of the normal edge is pointing, label the PPIR as a PIR and remove the other PPIR from the graph.

c For all oriented edges $D \rightarrow A$ in G for which only A belongs to the triangle, check whether A and D form a faithful triple and a v-structure with one of the two other nodes of the triangle (as in S3 of CPC). When testing the triple A, B and D, add $depset_{AB}$ to the conditioning set of the independence tests. If a v-structure is found, orient the two triangle edges containing A towards A and delete the third triangle edge if it is a PPIR.

to discover the v-structures. As a result in step S5a, the PPIR is removed and the correct structure is found. The correctness as well as the non-triviality follows then from the correctness and non-triviality of CPC, proven by (Ramsey et al., 2006).

2) No adjacency faithfulness:

We have to prove that no edge is deleted based on non-Markovian CIs, and that no mistakes are made during orientation.

2.1) No missing edges

A) Assume $X - Y$ in correct graph and $X \perp\!\!\!\perp Y$:

In step S2'[II]a, the algorithm looks for a variable T such that $X \not\perp\!\!\!\perp Y | T$. The existence of T follows from Minimality. Therefore the edge $X - Y$ will be replaced by a PPIR. Now, we show that this PPIR is not removed from the graph, which can only happen when there is a v-structure. If a PPIR would be removed based on the existence of a v-structure $X \rightarrow Z \leftarrow Y$ for some Z, then this indicates that the triangle faithfulness assumption is not satisfied. The removal of a PPIR in this case is an immediate consequence of the triangle faithfulness assumption which dictates that the direction of one of the arcs in the triangle imposes a v-structure. We do not orient v-structures containing equivalent edges, since a v-structure based on an equivalent edge which is not in the true graph could lead to erroneous deletion of a PIR when the equivalent edge appears in a triangle with the PIR.

B) Assume $X - Y$ in correct graph and $X \perp\!\!\!\perp Y | \boldsymbol{S}, \boldsymbol{S} \neq \emptyset$:

Take $Z \in \boldsymbol{S}$. Because of triangle faithfulness, Z cannot be adjacent to both X and Y, say it is not adjacent to X. Z can then be d-separated from X which gives

$X \perp\!\!\!\perp Z | Y \cup \boldsymbol{U}$. If $\boldsymbol{U}$ is not empty, from Assumption 5 it follows that $X \perp\!\!\!\perp Z | Y$. Edge $X - Y$ is not removed. If $\boldsymbol{U}$ is empty and $X \not\perp\!\!\!\perp Z | Y$, edge $X - Y$ will be removed temporarily. It will be added back when $X \perp\!\!\!\perp Z | Y \cup \boldsymbol{U}$ is discovered at a later stage.
C) Non-triviality is a direct consequence of the deletion of an edge $X - Y$ if for $\forall S \subseteq \{X, Y\}$ the independency $X \perp\!\!\!\perp Y | S$ holds.
2.2) Correct conservative orientation:
a) Orientation in the first step of the VCPC orientation phase (II') is only based on non-equivalent edges and non-PPIRs. So the correctness of CPC proves the correctness of these orientation steps in our algorithm.
b) We do not orient any edges based on equivalent edges.
c) Both S5b and S5c trigger when there is a known orientation of a normal edge inside a triangle with (a) PPIR(s) (S5b) or when the orientation of an edge in such a triangle can be inferred (S5c). A direct consequence of triangle faithfulness is that there is a v-structure at the node of the triple which has an incoming arrow. So the correctness of orientation follows from triangle faithfulness. ∎

5 Finite Sample Case

In this section we consider the finite sample case in which the independence oracle can make errors. Let's assume that the oracle for measuring CI is based on estimating the Dependency Strength (DS) and using a threshold for deciding independency. The smaller the sample, the more the estimated DS can deviate from the true value. A higher threshold is used for smaller sample sizes so that true independencies are not misclassified as dependencies. But this implies that the weaker a (conditional) dependency is, the more likely it gets misclassified as an independency. This is especially true as the DS becomes lower than the threshold. The oracle will only detect dependencies that are sufficiently strong. The following three cases should be considered.

5.1 Weak edges.

An edge $X - Y$ with a small $DS(X;Y)$ can still have a high $DS(X;Y|Z)$ when conditioned on one of the other parents, as is shown by the PIRs. A PIR still contains a lot of information, despite the marginal independence. Our extensions overcome missing PIRs or quasi-PIRs (edges that look like PIRs due to the finite sample size). On the other hand, if both $DS(X;Y)$ and $DS(X;Y|OthPa(X-Y))$ are small, we cannot overcome overlooking such edges, which we call *weak edges*. Limited data gives limited precision.

5.2 Near-to-unfaithfulness.

In general, dependencies with a low DS lead to near-to-unfaithful situations. Faithful distributions can come infinitely close to the unfaithful cases. This leads to the same CI patterns as in the unfaithful cases.

5.3 Weakening by conditioning.

A third way in which limited samples disrupt the learning is that an increased cardinality of the conditioning set reduces the robustness of most independence tests (Spirtes et al., 1993, p.116). We call this effect 'weakening by conditioning', which results in strict CIs not corresponding to minimal cut sets. They can be detected by the CIs corresponding to the conditions of Theorem 3.

6 Experimental results

To illustrate the adequacy of our extensions, simulations were performed on linear Gaussian and binary models. Experiments were performed on 100 randomly selected DAGs with d nodes and d edges, where d is randomly chosen between 5 and 25. For each such graph, a random structural equation model was constructed by selecting edge coefficients randomly uniformly from $[0.1, 1] \cup [-1, -0.1]$ and the variance of the disturbance terms was chosen randomly from $[0.01, 1]$. A random data set of 1000 cases was simulated for each of the models, to which the PC, CPC and VCPC algorithms were applied with depth 2 and significance level $\alpha = 0.05$ for each independence test based on Fisher's Z transformation of partial correlation. The output graph was compared to the Markov equivalence class (MEC) of the true DAG. Similar experiments were performed with Bayesian networks defined over a set of binary variables and randomly chosen conditional probabilities. The Chi-Square test was used as independence test.

The table on the next page shows the outcomes averaged over all experiments and relative to the number of nodes (percentages). Correct edges are the edges of the MEC of the true graph that appear as normal edges in the f-pattern. PPIRs and equivalent edges in the f-pattern are counted as ambiguous edges. False negative edges are edges in the MEC that do not appear in the f-pattern, not as a normal edge and not as an ambiguous edge. Weak edges are false negatives whose nodes are marginally independent and independent conditional on the other parents. False positive edges appear as normal edges in the f-pattern, but not in the MEC. If the nodes of a false positive are not d-connected in the MEC, they are classified as 'not connected'.

The learning performance of the orientation is evaluated by looking at edges appearing in both the MEC and the f-pattern. Edges having the same orientations in both are counted as correct orientations, when not oriented in

	PC	CPC	VCPC
Edges			
Correct	76.7	76.2	77.9
Ambiguous	0.0	0.0	74.4
False negatives	**23.2**	**23.8**	**8.1**
Weak	3.7	4.5	3.6
False positives	**4.1**	**4.3**	**11.8**
Not connected	2.9	3.1	9.3
Orientations			
Correct	25.3	29.8	36.6
Ambiguous	3.6	16.5	47.6
Wrong	**19.9**	**2.1**	**3.7**
False positives	**15.1**	**2.9**	**3.9**

both or only in f-pattern as ambiguous. Wrong orientations appear as oriented in both, but in the opposite direction. False positives are arrowheads appearing in the f-pattern but not in the MEC.

The results show that the difference between the PC and CPC lies clearly in the reduction of the false positive arrowheads, although we notice an increase in false negatives. The VCPC algorithm clearly reduces the number of false negative edges. If we consider that weak edges cannot be identified, the performance gain is even more drastic. By subtracting the number weak edges from the false negatives, the number of false negatives drops from 19.5%/19.3% for PC/CPC to 4.5% for VCPC. This drop is at the expense of ambiguous edges and more false positives. The latter can be explained by *accidental correlations*.

Accidental correlations lead to false negative independence tests - the oracle qualifies a Markovian CI as dependent due to accidentally-correlated data. VCPC is conservative about dependencies, it will not remove edges if there is no alternative path to explain a dependency. Take nodes that are not d-connected in the true graph ('not connected' in the table), but are accidently correlated. If this accidental correlation is above the threshold, the oracle will qualify it as a dependency. In the following steps, when conditioning happens on other variables, the weakening-by-conditioning effect will bring the measured dependency strength below the threshold and remove the 'accidental' edge. This happens with PC and CPC.

Finally, the experiments showed that the standard deviation for the false positive and negative edges is almost as high as the average, which points to a high performance fluctuation from one experiment to another.

7 Conclusions

We cannot rely on adjacency faithfulness when constructing robust learning algorithms. We showed that under triangle faithfulness, violations can be detected by two patterns: potential pseudo-independent relations (PPIRs) and equivalent edges. Based on both patterns, a set of DAGs can be identified that are indistinguishable from the perspective of the CIs. Just like the Conservative PC algorithm detects and treats failures of orientation-faithfulness, our Very Conservative PC algorithm detects violations of adjacency-faithfulness.

In the finite sample case, weak conditional dependencies can be wrongly classified as CIs by the oracle. This leads to near-to-unfaithful cases, weakening by conditioning and weak edges. The two first are treated, missing weak edges should be accepted. Since weak edges and triangle unfaithfulness cannot be detected, we believe that this analysis shows the natural bounds of what can reliably be learned under causal sufficiency.

References

Frederick Eberhardt. 2008. Almost optimal intervention sets for causal discovery. In *UAI*, pages 161–168.

Jan Lemeire. 2007. *Learning Causal Models of Multivariate Systems and the Value of it for the Performance Modeling of Computer Programs*. Ph.D. thesis, Vrije Universiteit Brussel.

Christopher Meek. 1995a. Causal inference and causal explanation with background knowledge. In *Procs of UAI-1995*, pages 403–41.

Christopher Meek. 1995b. Strong completeness and faithfulness in Bayesian networks. In *Procs of UAI-1995*, pages 411–418.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, Morgan Kaufman Publishers.

Joseph Ramsey, Jiji Zhang, and Peter Spirtes. 2006. Adjacency-faithfulness and conservative causal inference. In *Procs of UAI-2006*, pages 401–408.

Peter Spirtes, Clark Glymour, and Richard Scheines. 1993. *Causation, Prediction, and Search*. Springer Verlag, 2nd edition.

Harald Steck and Volker Tresp. 1999. Bayesian belief networks for data mining. In *Procs of the 2nd Workshop on Data Mining und Data Warehousing*.

Yang Xiang, S. K. Wong, and N. Cercone. 1996. Critical remarks on single link search in learning belief networks. In *Procs of UAI-1996*, pages 564–571.

Jiji Zhang and Peter Spirtes. 2007. Detection of unfaithfulness and robust causal inference. In *Procs of the LSE-Pitt Conference: Confirmation, Induction and Science, London*.

Scaling Up MAP Search in Bayesian Networks Using External Memory

Heejin Lim, Changhe Yuan, and Eric A. Hansen
Department of Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762

Abstract

State-of-the-art exact algorithms for solving the MAP problem in Bayesian networks use depth-first branch-and-bound search with bounds computed by evaluating a join tree. Although this approach is effective, it can fail if the join tree is too large to fit in RAM. We describe an external-memory MAP search algorithm that stores much of the join tree on disk, keeping the parts of the join tree in RAM that are needed to compute bounds for the current search nodes, and using heuristics to decide which parts of the join tree to write to disk when RAM is full. Preliminary results show that this approach improves the scalability of exact MAP search algorithms.

1 Introduction

State-of-the-art exact MAP algorithms for Bayesian networks use depth-first branch and bound (DF-BnB) search, and prune the search tree using bounds that are computed by evaluating a join tree (Park and Darwiche, 2003; Yuan and Hansen, 2009) or an arithmetic circuit (Huang et al., 2006). For large and complex Bayesian networks, however, the join tree or arithmetic circuit can be too large to fit in RAM, limiting scalability. In this paper, we focus on the approach that uses a join tree to compute bounds. The memory required to store the join tree is exponential in the treewidth of the Bayesian network.

We describe how to improve the scalability of a MAP search algorithm that evaluates a join tree to compute bounds by using external memory to store the join tree when it is too large to fit in RAM. The efficiency of this approach depends on the heuristics used to decide which parts of the join tree to write to disk when RAM is full. Our study shows that commonly used heuristics for external-memory algorithms, such as *least recently used* (LRU) and *least frequently used* (LFU), do not always perform well in MAP search. We introduce new heuristics that take into account the unique characteristics of the search algorithm. Preliminary results show that the approach can solve MAP problems that could not previously be solved due to memory limitations.

2 Background

We begin with a brief review of the MAP problem and algorithms for solving the MAP problem using branch-and-bound search.

2.1 The MAP problem

The Maximum a Posteriori assignment problem (MAP) is defined as follows. Let $\mathbf{M}$ be a set of *explanatory variables* in a Bayesian network; from now on, we call these the *MAP variables*. Let $\mathbf{E}$ be a set of *evidence variables* whose states have been observed. The remaining variables, denoted $\mathbf{S}$, are variables for which the states are unknown and not of interest. Given an assignment $\mathbf{e}$ for the variables $\mathbf{E}$, the MAP problem is to find an assignment $\mathbf{m}$ for the variables $\mathbf{M}$ that maximizes the joint probability $P(\mathbf{m}, \mathbf{e})$ (or, equivalently, the conditional probability $P(\mathbf{m}|\mathbf{e})$). Formally,

$$\hat{\mathbf{m}}_{MAP} = \arg\max_{\mathbf{M}} \sum_{\mathbf{S}} P(\mathbf{M}, \mathbf{S}, \mathbf{E} = \mathbf{e}) \ , \quad (1)$$

where $P(\mathbf{M}, \mathbf{S}, \mathbf{E} = \mathbf{e})$ is the joint probability distribution of the network given the assignment $\mathbf{e}$.

2.2 Join tree upper bound

In Equation (1), the maximization and summation operators are applied to different sets of variables. The MAP variables in $\mathbf{M}$ can be maximized in different orders, and the variables in $\mathbf{S}$ can be summed

Pp. 177–185 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

out in different orders, without affecting the result. But the summations and maximizations are not commutable. As a result, variable elimination-based methods for solving MAP have a complexity that depends on the *constrained treewidth* of the network, and they are typically infeasible because they require too much memory.

If the ordering among the summations and maximizations is relaxed, however, an upper bound on the probability of a MAP solution is computed. The following theorem is due to Park and Darwiche (2003).

Theorem 1. *Let $\phi(\mathbf{M}, \mathbf{S}, \mathbf{Z})$ be a potential over the disjoint variable sets $\mathbf{M}$, $\mathbf{S}$, and $\mathbf{Z}$. For any instantiation $\mathbf{z}$ of $\mathbf{Z}$, the following inequality holds:*

$$\sum_S \max_M \phi(\mathbf{M}, \mathbf{S}, \mathbf{Z} = \mathbf{z}) \geq \max_M \sum_S \phi(\mathbf{M}, \mathbf{S}, \mathbf{Z} = \mathbf{z}) .$$

Based on this result, Park and Darwiche (2003) compute upper bounds for the MAP problem using the join tree algorithm, but with redefined messages. Each message is computed such that variables in $\mathbf{S}$ are summed over before MAP variables are maximized.

2.3 Solving MAP using DFBnB

Park and Darwiche (2003) use the join tree upper bound in a depth-first branch-and-bound (DFBnB) search algorithm to solve the MAP problem. Since a full evaluation of the join tree computes simultaneous upper bounds for all MAP variables, Park and Darwiche use dynamic variable ordering to speed up their search algorithm. Yuan and Hansen (2009) observe that, when a static variable ordering is used, it is only necessary to compute bounds for the next MAP variable to be instantiated at each step, and this only requires evaluating a small part of the join tree. They use a static ordering of MAP variables that is created from a post-order traversal of the MAP variables in the join tree. With this static ordering, the upper bounds needed for the next instantiating variable(s) in MAP search can be computed incrementally by message passing along a limited and fixed path in the join tree. During forward traversal of a branch of a search tree, it is only necessary to perform message passing once along this path in the join tree, broken up into separate steps

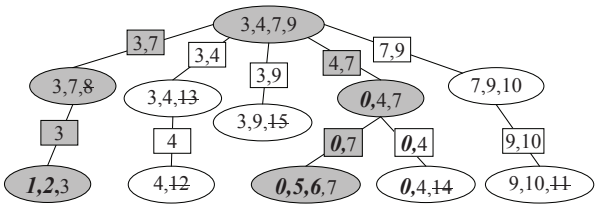


Figure 1: Example of a join tree for upper bound computation. The shaded nodes are nodes of the join tree that contain MAP variables.

for each instantiating variable. To allow efficient backtracking, the clique and separator potentials of the join tree that are changed during bounds computation are cached in the order that they are changed. During backtracking, the cached potentials can be used to efficiently restore the join tree to its previous state before one or more MAP variables were instantiated. The readers are referred to (Yuan and Hansen, 2009) for more details of the algorithm.

We illustrate the idea with an example. Figure 1 shows the join tree of a Bayesian network based on the Hugin architecture (Jensen et al., 1990). The numbers are the indices of distinct variables in the network, and the numbers in bold-face italics represent the MAP variables, which are 0, 1, 2, 5, and 6. Let the static search ordering of the MAP variables be: 1,2,0,5,6. After MAP variables 1 and 2 are instantiated, their values are entered as evidence to clique $\{1, 2, 3\}$. Messages can then be sent to the other parts of the join tree to get upper bounds for the remaining MAP variables. However, since the next variable in the static ordering is 0, it is only necessary to send messages along the shaded path from clique $\{1, 2, 3\}$ to clique $\{0, 4, 7\}$. None of the other parts of the join tree need to be involved in the propagation. The path has a new set of clique and separator potentials as a result of the message propagation. The old potentials are cached before they are overwritten with the new potentials. If all the search nodes after instantiating variable 0 can be pruned using upper bounds, the search algorithm backtracks to the parent search node and retracts the join tree to the previous state. This can be achieved by simply restoring the cached potentials in reverse order and rolling back the changes.

3 External-memory MAP Search

The MAP search algorithms of Park and Darwiche (2003) and Yuan and Hansen (2009) use join tree evaluation to compute bounds for a depth-first branch-and-bound search. We use the Yuan and Hansen algorithm as the basis for our external-memory algorithm, in part because it has been shown to be more efficient than the Park and Darwiche algorithm when their internal-memory versions are compared, and in part because its incremental approach is easier to convert to an efficient external-memory algorithm. Each time the Park and Darwiche algorithm computes bounds for a node of the search tree, it must perform a full join tree evaluation; this could require copying the entire join tree into RAM for each search node, incurring a large amount of disk I/O. By contrast, each time the Yuan and Hansen algorithm computes bounds for a node of the search tree, it only needs to keep a small part of the join tree in RAM. As we will see, the locality that it exploits for incremental message propagation is also exploited by an external-memory version of the algorithm to minimize disk I/O.

3.1 Memory architecture

Figure 2 illustrates how our algorithm uses RAM and disk. We call the jointree without potentials the *skeleton*. The skeleton is typically small and always resides in RAM, as shown in Figure 2. Each clique of the skeleton has one *main pointer* that points to the clique's potential, which we call the *main potential*. The clique may also have one or more *cache pointers* that point to cached copies of the potential. The basic idea of the algorithm is to store some of the potentials in RAM and some on disk. When a potential is stored on disk, it is stored in a file named by a unique *tag* assigned to the potential. In Figure 2, arc 1 shows an example of a main pointer, and arc 2 shows an example of a cache pointer.

All computation on the jointree is based on the main potentials. The other potentials are only for caching purposes. A potential needed for computation should reside in RAM. When necessary, we can *write* a potential in RAM to a disk file named after the potential's tag, in order to free up space in RAM; we can also use a tag to find the appropriate disk file to *read* the potential back into RAM.

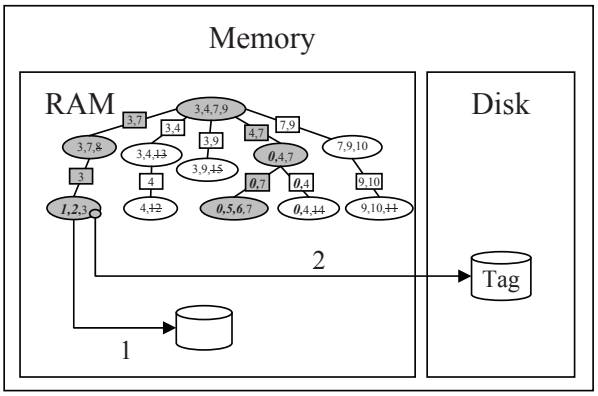


Figure 2: Memory architecture of the algorithm.

3.2 Initialize an upper-bound join tree

In (Yuan and Hansen, 2009), we initialize an upper-bound jointree for a Bayesian network completely in RAM. We first create its skeleton. We then initialize all the clique potentials with appropriate conditional probability tables of the Bayesian network. After entering evidence to the jointree, a full join tree propagation is performed to finish initializing the upper-bound join tree. A full join tree propagation involves two phases: *collect* and *distribute*. In the collect phase, all cliques send messages to their parents by combining messages sent from children cliques. After the root receives all messages from its children, the distribute phase starts, in which all cliques send messages to their children by combining messages from parents.

Several preprocessing methods can be used to reduce the size of an upper-bound jointree in order to delay the use of external memory. First, we use relevance reasoning (Lin and Druzdzel, 1997) to preprocess a Bayesian network based on the evidence and target variables of a MAP problem. This step reduces the size of the network by removing irrelevant variables such as barren variables. Second, we create an upper-bound join tree that is as small as possible without considering the quality of the bounds. Third, we can release clique potentials that are only needed during join tree initialization, and not during MAP search. By *releasing*, we mean that the potentials are deleted and their pointers are set to NULL. With incremental join tree bounds, only part of the join tree is involved in message propagation during MAP search. In Figure 1, the sec-

ond left-most branch has no MAP variables. Once a clique on this branch has sent a message to its parent, the clique can be released from memory. In fact, all non-shaded parts of the join tree in Figure 1 can be released before MAP search. Doing so not only postpones the use of external memory, it improves the efficiency of the algorithm, because there is no need to send messages to these parts of the join tree during the distribute phase.

Nevertheless, the final upper-bound join tree may still be too large to fit in RAM. In that case, we must construct the join tree incrementally and store parts of it on disk. We do so as follows. After constructing the skeleton, we initialize the jointree by *interleaving* potential initialization and message propagation via a *left-to-right, leaf-to-root* traversal of the jointree. For Figure 1, we would start by initializing the potential of clique $\{1, 2, 3\}$ in the left-most branch. We then initialize the potential for separator $\{3\}$ by calculating the message to be sent from $\{1, 2, 3\}$ to $\{3, 7, 8\}$. After that, we initialize the potential of clique $\{3, 7, 8\}$ and combine the message stored in separator $\{3\}$. To avoid exhausting RAM, we estimate the amount of additional RAM needed for each step of the algorithm and check if the increase is larger than the amount of available RAM. If there is not sufficient RAM available, we have to write to disk some cliques or separators that are already constructed and reside in RAM. In a later section of the paper, we introduce several heuristics for selecting which cliques and separators to write to disk. For now, it suffices to say that enough RAM will be freed so that the current step can be executed. We can use the above incremental scheme to complete the collect phase.

The collect phase traverses a join tree from left to right. After the collection phase finishes, the cliques stored on disk most likely come from the leftmost branches. Since the distribute phase may need to restore some clique potentials from disk to RAM, the distribute phase uses a *right-to-left, root-to-leaf* traversal of the join tree so as to use clique potentials that currently reside in RAM first. The leftmost branches are not read from disk until they are needed. After the distribute phase finishes, cliques from the rightmost branches are swapped out to disk. This improves the performance of MAP search because the search starts from the leftmost branches as well.

We may still be able to release part of the join tree during the distribute phase. When all of the MAP variables are in one branch of the join tree, we can release the root and its immediate successor cliques if they do not contain MAP variables.

Finally, calculating a message requires the potentials of at least one clique and one separator to be in RAM at the same time. For example, we need both clique $\{1, 2, 3\}$ and separator $\{3\}$ to be able to compute the message to be sent to $\{3, 7, 8\}$. Therefore, our method has a *minimal* memory requirement that is equal to the largest total size of any neighboring pair of clique and separator.

3.3 MAP search using external memory

Once the upper-bound join tree is initialized, we start the MAP search. At each search step, we need to use the join tree to compute the search bounds, which requires message propagation on the jointree. For each message propagation step, we check whether or not the potentials we need reside in RAM. If not, we check whether there is enough RAM available for reading them back in RAM from disk. If necessary, we use the heuristics described in Section 3.4 to select potentials to write from RAM to disk in order to free up enough RAM to continue. The strategy of using external memory is similar to the strategy used in the join tree initialization phase. There are, however, some important differences that warrant discussion.

For efficient backtracking, the Yuan and Hansen algorithm caches and restores potentials during the MAP search. The need for caching makes using external memory slightly more complicated. New strategies are needed, both during *forward search* and *backtracking*.

During *forward search*, we need to cache a potential before we set the state of a newly instantiated MAP variable as evidence to a clique potential, and before we update a clique or separator potential using incoming messages. There are two possibilities. One is that the potential to be cached is in RAM. If enough additional RAM is available, we make a copy in RAM immediately. If not, we free up space by writing some cliques from RAM to disk before making the copy in RAM. The second possibility is that the potential is on disk. Since we need the po-

tential in the next operation, we read a copy of the potential from disk to RAM and swap the main and cache pointers so that the main pointer points to the copy in RAM.

When *backtracking*, we need to restore the join tree to a previous state by restoring some cached potentials. A potential and its cached copy can be in RAM and/or disk. No matter where they are, we simply delete the current main potential and redirect the main pointer to point to the cached copy. Therefore, no disk I/O is needed during backtracking. We only read potentials from disk to RAM when they are needed during forward search.

The strategies described above allow a potential that is once written to disk to remain on disk and to be repeatedly used until it is not needed anymore. This helps to limit the number of times the same potential is written to disk or read from disk.

3.4 Heuristics

It is critical for our MAP algorithm to have a good heuristic for selecting which cliques and separators to write to disk when RAM is full, in order to keep the amount of disk I/O as low as possible.

Commonly-used heuristics include storing the *least recently used* (LRU) or the *least frequently used* (LFU) data in external memory. We implemented both and found that the LRU heuristic outperforms the LFU heuristic because message propagation follows a fixed post-traversal order of the join tree. We also tested two other heuristics. One heuristic that we call *largest first* (LF) selects the largest cliques to store in external memory. The other heuristic that we call *largest but least frequently used* (LLF) integrates the LF and LFU heuristics. LLF selects the largest cliques, but decreases the priority of a clique if it is selected too often. More formally, we use the following formula to order the candidate cliques,

$$priority = \frac{size}{\#I/O}, \tag{2}$$

where $size$ is the size of a clique and $\#I/O$ is the number of times the clique is written to disk or read from disk. The clique that has the largest $priority$ value is selected as the next clique to write to disk.

4 Empirical evaluation

We tested our external-memory algorithm (DFBnB+EM) by comparing it to the original internal-memory MAP algorithm (DFBnB) developed by Yuan and Hansen (2009). Experiments were performed on a 2.1GHz processor with 4GB of RAM running a 32-bit version of Windows Vista. The user is only allowed to use 2GB of the memory address in a 32-bit environment. The magnetic disk used for external memory operates at 5400RPM and its interface is SATA2 (Serial Advanced Technology Attachment).

4.1 Benchmarks and experimental design

Algorithm performance was tested on a set of benchmark Bayesian networks. Two of the networks, BN-43 and BN-44, are from the UAI-06 inference competition. For these two networks, the MAP problem cannot be solved without using disk. This shows that the new algorithm increases the range of problems for which the MAP problem can be solved exactly.

The other networks have been previously used to test the internal-memory version of the MAP search algorithm. They allow us to compare the performance of the internal-memory MAP-search algorithm to the external-memory algorithm under artificial memory limits. For each Bayesian network, we set the *low* memory limit based on the minimal memory requirement for the join tree of the network. We set the *high* memory limit by averaging the low memory limit and the total memory used by the internal-memory MAP search algorithm. For the BN-43 and BN-44 networks, we set the low memory limit in the same way but set the high memory limit to be all available RAM.

Since the artificial memory limits are set based on the join tree size, they do not consider other memory requirements of the MAP algorithm, such as memory incurred during search. Even when the initial join tree fits in physical memory, caching potentials during MAP search can significantly increase the amount of memory needed. Yuan and Hansen (2009) found that the increase ranges from slightly larger to several times larger. Therefore, MAP search may still fail if external memory is not used. Finally, the Windows OS typically allocates

Network	DFBnB							
	#Nodes	#Backtracks	Build(ms)	Search(ms)	Largest clique	Jointree	Avg memory	Max memory
Hailfinder	9,902	61,721	15	457	26K	98K	231K	231K
Water	5	14	1,683	372	41M	70M	126M	140M
Munin4	195	104	57,332	567	8M	155M	180M	408M
Barley	282	2,524	8,949	62,858	100M	230M	345M	475M
Mildew	1,135	6,102	5,490	13,684	43M	104M	166M	167M
Andes	124,971	975,861	743	80,522	2M	5M	14M	14M
Pigs	75,627	660,986	2,313	150,206	4M	11M	32M	33M
Diabetes	11,783	161,262	7,629	204,837	2M	89M	144M	150M
BN-43	5,520	54,700	-	-	512M	1,008M	-	-
BN-44	1,292	16,276	-	-	258M	890M	-	-

Table 1: Results for the DFBnB algorithm of Yuan and Hansen (2009). '#Nodes' is the number of search nodes. '#Backtracks' is the number of backtracking steps. 'Largest clique' is the largest clique size, and 'Jointree' is the jointree size. 'Avg memory' is the average total memory of the test cases, while 'Max memory' is the maximum total memory. 'K' means kilobytes, and 'M' megabytes. Since the internal-memory algorithm could not solve the MAP problem for networks BN-43 and BN-44, the partial results in the table are from the external-memory algorithm.

a significant portion of memory to service applications. Therefore, not all RAM is available for use by the MAP algorithm.

For each of the benchmark Bayesian networks, we generated 10 random test cases with as many root nodes as MAP variables and with all leaf nodes as evidence variables so that they are solvable within reasonable time. Tables 1 and 2 report the average results for these test cases.

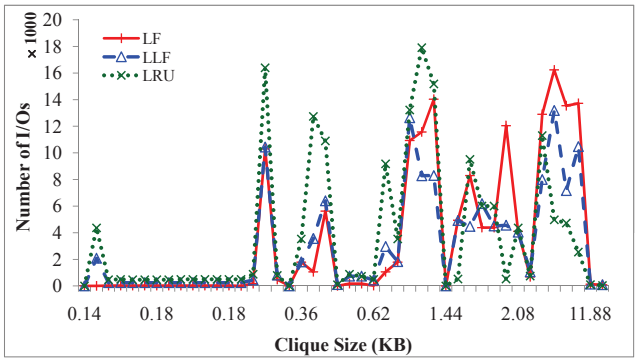


Figure 3: Number of times chosen for I/O (in thousands) versus clique size for different heuristics on the Hailfinder network.

4.2 Analysis of results

Table 1 shows the results for the DFBnB algorithm and Table 2 shows the results for the DFBnB+EM algorithm. However, note that the numbers of search nodes (#Nodes) and backtracks (#Backtracks) shown in Table 1, as well as the size of the largest clique and the size of the join tree, are the same for both algorithms.

Unsurprisingly, there is more disk I/O (both reads and writes) when there is more backtracking. But interestingly, the amount of data read from disk to RAM is often larger than the amount of data written from RAM to disk. During backtracking, we need to restore some cached potentials. If they are in external memory, we do not need to immediately read them from disk to RAM; we just pass their tags to the potential pointers. When forward search resumes, copies are then read from disk to RAM for message propagation, and main and cache pointers are swapped so that the cache pointer points to the external-memory copy. No write is necessary during this process, unless memory is running low. This is desirable because, for the same amount of data, writing to disk takes more time than reading from disk. This can be explained as follows. When there is not much backtracking, however, as for the Munin network, it is possible for more data to be written to disk than read from disk.

Table 2 compares the performance of the three heuristics described in Section 3.4: LRU, LF, and LLF. The results indicate that LRU typically incurs the least amount of I/O. A join tree typically has smaller leaf cliques and larger inner cliques, and inner cliques are accessed more often during back-

Network	DFBnB+EM							
	Memory limit	Heuristics	Jointree building			MAP search		
			Time	Write	Read	Time	Write	Read
Hailfinder	40K	LF	836	303K	227K	147,730	58M	97M
		LLF	881	303K	227K	**91,539**	33M	73M
		LRU	1,218	302K	209K	156,263	**9M**	**62M**
	135K	LF	16	0K	0K	**900**	507K	**8M**
		LLF	19	0K	0K	1,915	298K	20M
		LRU	19	0K	0K	3,293	**267K**	27M
Water	55M	LF	2,951	180M	164M	**1,767**	**223M**	**234M**
		LLF	2,972	180M	164M	2,199	**223M**	**234M**
		LRU	3,187	186M	168M	2,034	237M	240M
	90M	LF	1,758	0M	0M	**957**	93M	**93M**
		LLF	1,766	0M	0M	1,108	**62M**	132M
		LRU	1,778	0M	0M	1,296	91M	151M
Munin4	10M	LF	67,879	697M	534M	**2,940**	334M	333M
		LLF	69,180	697M	534M	8,311	333M	332M
		LRU	81,317	570M	410M	20,562	**239M**	**234M**
	95M	LF	59,037	186M	123M	2,305	102M	93M
		LLF	59,870	248M	180M	**2,138**	114M	98M
		LRU	74,017	281M	206M	10,581	**84M**	**66M**
Barley	150M	LF	11,906	363M	293M	266,415	24G	**30G**
		LLF	11,605	363M	293M	**259,678**	**22G**	**30G**
		LRU	11,522	313M	275M	291,873	24G	31G
	250M	LF	8,859	0M	0M	229,783	21G	27G
		LLF	8,923	0M	0M	113,009	6G	17G
		LRU	8,869	0M	0M	**88,168**	**249M**	**15G**
Mildew	50M	LF	8,065	335M	272M	46,589	5G	7G
		LLF	7,859	335M	272M	47,537	5G	7G
		LRU	8,331	344M	280M	**40,764**	**2G**	**4G**
	110M	LF	5,643	0M	0M	16,395	**55M**	2G
		LLF	5,588	0M	0M	**16,373**	**55M**	2G
		LRU	5,680	0M	0M	18,595	66M	**606M**
Andes	5M	LF	782	2M	1M	139,772	3G	12G
		LLF	775	1M	1M	**120,408**	**977M**	9G
		LRU	818	0M	0M	362,868	1G	**8G**
	10M	LF	772	0M	0M	**84,679**	319M	**1G**
		LLF	757	0M	0M	87,456	**13M**	5G
		LRU	765	0M	0M	175,501	51M	7G
Pigs	10M	LF	2,528	10M	8M	877,444	77G	86G
		LLF	2,558	15M	12M	**399,216**	23G	47G
		LRU	6,251	9M	6M	1,866,499	**8G**	**42G**
	20M	LF	2,384	0M	0M	**184,497**	523M	29G
		LLF	2,363	0M	0M	197,263	**368M**	27G
		LRU	2,443	0M	0M	1,163,290	3G	**23G**
Diabetes	5M	LF	10,821	155M	115M	3,445,326	115G	124G
		LLF	11,213	159M	119M	**2,275,228**	62G	77G
		LRU	11,475	146M	106M	2,661,619	**44G**	**62G**
	75M	LF	7,694	0M	0M	238,486	147M	26G
		LLF	7,680	0M	0M	238,031	51M	30G
		LRU	7,682	0M	0M	**227,471**	**34M**	**20G**
BN-43	600M	LF	81,276	4G	4G	1,760,086	41G	46G
		LLF	82,602	4G	4G	1,804,655	41G	46G
		LRU	71,151	3G	3G	**1,114,261**	**24G**	**29G**
	Available	LF	13,962	0M	0M	514,972	10G	20G
		LLF	13,775	0M	0M	481,041	**5G**	16G
		LRU	13,962	0M	0M	**451,230**	**5G**	**15G**
BN-44	350M	LF	70,731	4G	3G	3,513,469	251G	254G
		LLF	69,006	4G	3G	3,472,347	239G	245G
		LRU	77,144	3G	3G	**3,219,000**	**208G**	**214G**
	Available	LF	18,426	0M	0M	537,318	16G	64G
		LLF	17,984	0M	0M	**407,260**	**1G**	**50G**
		LRU	18,225	0M	0M	510,685	12G	56G

Table 2: Results of external-memory MAP search. Running time is measured in milliseconds. 'K' means kilobytes, 'M' means megabytes, and 'G' means gigabytes.

tracking search. Therefore, LRU tends to select smaller cliques to write to disk. By contrast, LF and LLF select larger cliques to write to disk. This result is clearer in Figure 3, which plots the number of disk I/O operations versus clique size in solving the MAP problem for the Hailfinder network. The two largest cliques are not selected often by LF and LLF because they are stored in external memory early in the search, and they remain there most of the time. They are only occasionally copied from disk back to RAM during backtracking.

However, the amount of disk I/O is not directly proportional to running time. Although LRU has the least amount of I/O and is faster than LF and LLF on some networks, it can be much slower on some other networks, such as Munin4 and Pigs. One reason for this is the small cliques chosen by LRU may not completely fill the I/O buffers, leading to more I/O operations. If all cliques are large, as they are for BN-43 and BN-44, LRU may perform better.

The LF heuristic may repeatedly select the same large cliques to write to disk because it only considers clique size. The LLF heuristic decreases the frequency with which the same large cliques are selected by prioritizing cliques based on dividing clique size by the number of times the clique has been written to or read from disk. Results show that when there is a lot of backtracking and not much RAM available, LLF can be significantly faster than LF. This is illustrated by the results for Hailfinder and Barley. Otherwise, LF and LLF demonstrate similar behavior in most cases.

We can use the ratio of the size of the join tree to the size of the largest clique to estimate the improvement in scalability from using our external-memory algorithm. For the benchmark networks, the ratio ranges from several times to around 45 times. The ratio is even higher if we take into account all memory used by the original MAP algorithm, since caching potentials can increase the size of a join tree.

5 Conclusion

We have introduced an external-memory approach to scaling up a depth-first branch-and-bound algorithm for solving the MAP problem that uses incremental join tree bounds (Yuan and Hansen, 2009). Our results show that the external-memory approach improves scalability and allows MAP problems to be solved exactly that could not be solved before due to memory limitations.

The minimum memory requirement of our algorithm is the amount of memory needed to store any neighboring pair of clique and separator. We plan to address this limitation by allowing just part of a clique to be stored in RAM while the rest is stored on disk (Kask et al., 2010). We will try to develop better heuristics for selecting potentials to write to external memory. Although we already use relevance reasoning to exploit evidence-based independence, we will consider whether the lazy propagation architecture proposed in (Madsen and Jensen, 1999) can improve the efficiency of our algorithm.

Acknowledgments This research was supported by NSF grants IIS-0953723, EPS-0903787, and IIS-0812558.

References

Jinbo Huang, Mark Chavira, and Adnan Darwiche. 2006. Solving map exactly by searching on compiled arithmetic circuits. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, page 143148.

Finn V. Jensen, Steffen L. Lauritzen, and Kristian G. Olesen. 1990. Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly*, 4:269–282.

Kalev Kask, Rina Dechter, and Andrew Gelfand. 2010. BEEM: Bucket elimination with external memory. In *Proceedings of The 26th Conference on Uncertainty in Artificial Intelligence (UAI-10)*, AUAI Press Corvallis, Oregon.

Yan Lin and Marek J. Druzdzel. 1997. Computational advantages of relevance reasoning in Bayesian belief networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 342–350, San Francisco, CA. Morgan Kaufmann Publishers, Inc.

Anders L. Madsen and Finn V. Jensen. 1999. Lazy propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1-2):203–245.

James D. Park and Adnan Darwiche. 2003. Solving MAP exactly using systematic search. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI–03)*, pages 459–468, Morgan Kaufmann Publishers San Francisco, California.

Changhe Yuan and Eric A. Hansen. 2009. Efficient computation of jointree bounds for systematic MAP search. In *Proceedings of 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1982–1989, Pasadena, CA.

Modelling the Interactions between Discrete and Continuous Causal Factors in Bayesian Networks

Peter J.F. Lucas and Arjen Hommersom
Radboud University Nijmegen,
Institute for Computing and Information Sciences, The Netherlands
Email: {peterl,arjenh}@cs.ru.nl

Abstract

The theory of causal independence is frequently used to facilitate the assessment of the probabilistic parameters of discrete probability distributions of complex Bayesian networks. Although it is possible to include continuous parameters in Bayesian networks as well, such parameters could not, so far, be modelled by means of causal independence theory, as a theory of continuous causal independence was not available. In this paper, such a theory is developed and generalised such that it allows merging continuous with discrete parameters based on the characteristics of the problem at hand. This new theory is based on the discovered relationship between the theory of causal independence and convolution in probability theory, discussed for the first time in this paper. It is also illustrated how this new theory can be used in connection with special probability distributions.

1 Introduction

One of the major challenges in building Bayesian networks is to estimate the associated probabilistic parameters. As these parameters of a Bayesian network have the form of conditional probability distributions $P(E \mid C_1, \ldots, C_n)$, it has been beneficial to look upon the interaction between the associated random variables E, $C_1, \ldots, C_n$ as the interactions between *causes* C_k and an *effect* E. This insight has driven much of the early work (Pearl, 1988), and is still one of the main principles used to construct Bayesian networks for actual problems.

Causal principles have also been exploited in situations where the number of causes n becomes large, as the number of parameters needed to assess a family of conditional probability distributions for a variable E grows exponentially with the number of its causes. The theory of causal independence is frequently used in such situations, basically to decompose a probability table in terms of a small number of causal factors (Henrion, 1989; Pearl, 1988; Heckerman and Breese, 1996). However, so far this theory was restricted to the modelling of *discrete* probability distributions, where in particular three types of interaction are in frequent use: the noisy-OR and the noisy-MAX—in both cases, the interaction among variables is being modelled as disjunctive (Díez, 1993; Henrion, 1989; Pearl, 1988)—and the noisy-AND. Interactions among *continuous* cause variables are usually modelled by statistical techniques such as logistic regression and probit regression, typically by using iterative numerical methods that estimate the weight parameters by maximising the likelihood of the data given the model (Bishop, 2006). Clearly, these regression models resist manual construction based on a solid understanding of a problem domain; the fact that Bayesian networks can be constructed using a mixture of background knowledge and data, depending on the availability of knowledge and data of the problem at hand, is seen as one of the key benefits of the technique. Moreover, it is not possible to combine regression models with discrete causal independence models.

In this paper, a new framework of causal independence modelling is proposed. It builds upon the link we discovered between the theory of causal independence and the convolution theorem of probability theory. The framework is developed by generalising this theorem into an algebra that supports the modelling of interactions, whether discrete, continuous, or both, in a meaningful way.

Pp. 185–193 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

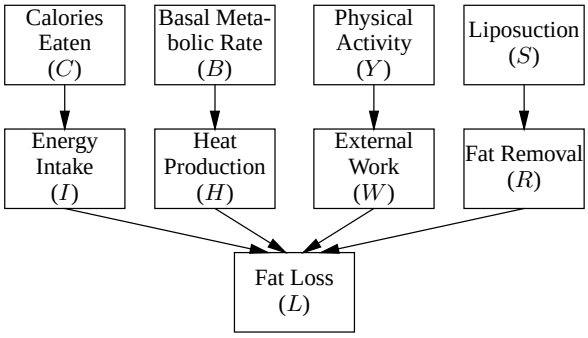


Figure 1: Causal factors that affect fat loss in humans.

2 Motivating Example

In biomedical modelling one often has to deal with a mixture of discrete and continuous causes that give rise to an effect. For example, the amount of *fat storage* in the human body is determined by the *energy balance*, i.e., the balance between energy intake and expenditure. A decrease in fat storage usually occurs whenever the energy intake is smaller than the energy expenditure. The energy expenditure is determined by the internal heat produced, which is mainly the basal metabolic rate (BMR), plus external work estimated by physical activity. Besides altering the energy balance, the storage can be decreased by means of *liposuction*. The energy variables are naturally represented as continuous variables, whereas 'Liposuction' is discrete.

The causal model is presented in Figure 1 and the conditional probability distributions of fat loss are represented by: $P(L \mid C, B, Y, S)$. Somehow this distribution must be determined by the interaction between the intermediate causal variables concerned, expressed by $A \equiv (I \leq (H + W))$ (energy intake is less than or equal to heat production plus external work), with A standing for an appropriate energy balance. Furthermore, the binary (Boolean) effect variable fat loss L is defined as $L \equiv (A \vee R)$ (fat loss L is due to a change in the energy balance A or fat removal R). The techniques developed in this paper will allow one to exploit such information in building a Bayesian network.

3 Preliminaries

This section provides a review of the basics underlying the research of this paper.

3.1 Probability theory and Bayesian networks

In this paper we are concerned with both discrete and continuous probability distributions P, defined in terms functions f, called the probability mass function for the discrete case and density function for the continuous case. Associated with a mass and density function, respectively, are distribution functions, denoted by F. Random variables are denoted by upper case, e.g., X, I etc. Instead of $X = x$ we will frequently write simply x. This is also the notation used to vary over values in summation and integration and to indicate that a binary variable X has the value 'true'. The value 'false' of a binary variable X is denoted by $\bar{x}$. Finally, free variables are denoted by uppercase, e.g., X.

A *Bayesian network* is a concise representation of a joint probability distribution on a set of random variables (Pearl, 1988). It consists of an acyclic directed graph $G = (\mathcal{V}, \mathcal{A})$, where each node $V \in \mathcal{V}$ corresponds to a random variable and $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of arcs. The absence of arcs in the graph G models independences between the represented variables. In this paper, we give an arc $V \rightarrow V'$ a causal reading: the arc's direction marks V' as the *effect* of the *cause* V. In the following, causes will often be denoted by C_i and their associated effect variable by E.

Associated with the qualitative part of a Bayesian network are numerical parameters from the encoded probability distribution. With each variable V in the graph is associated a set of *conditional probability distributions* $P(V \mid \pi(V))$, describing the joint influence of values for the parents $\pi(V)$ of V on the probabilities of the variable V's values. These sets of probabilities constitute the quantitative part of the network. A Bayesian network represents a joint probability distribution of its variables and thus provides for computing any probability of interest.

3.2 Causal modelling

One popular way to specify interactions among statistical variables in a compact fashion is offered by the notion of *causal independence* (Heckerman and Breese, 1996). The global structure of a causal-independence model is shown in Figure 2; it expresses the idea that causes $C = (C_1, \ldots, C_n)$ influence a given common effect E through interme-

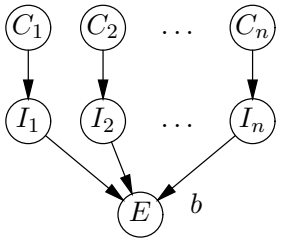


Figure 2: Causal independence model.

diate variables $I = (I_1, \ldots, I_n)$ and a Boolean, or Boolean-valued, function b, called the *interaction function*. The influence of each cause C_k on the common effect E is independent of each other cause C_j, $j \neq k$. The function b represents in which way the intermediate effects I_k, and indirectly also the causes C_k, interact to yield the final effect E. Hence, this function b is defined in such way that when a relationship, as modelled by the function b, between I_k, $k = 1, \ldots, n$, and $E = 1$ (true) is satisfied, then it holds that $b(I_1, \ldots, I_n) = 1$, denoted by $b(I_1, \ldots, I_n) = e$.

The conditional probability of the occurrence of the effect E given the causes $C_1, \ldots, C_n$, can be obtained from the conditional probabilities $P(I_k \mid C_k)$ as follows:

$$P_b(e \mid C_1, \ldots, C_n) = \sum_{b(i_1,\ldots,i_n)=e} \prod_{k=1}^{n} P(i_k \mid C_k) \quad (1)$$

Formula (1) is practically speaking not very useful, because the size of the specification of the function b is exponential in the number of its arguments. The resulting probability distribution is therefore in general computationally intractable, both in terms of space and time requirements. An important subclass of causal independence models, however, is formed by models in which the deterministic function b can be defined in terms of separate binary functions g_k, also denoted by $g_k(I_k, I_{k+1})$. Such causal independence models have been called *decomposable* causal independence models (Heckerman and Breese, 1996); these models are of significant practical importance. Often, all functions $g_k(I_k, I_{k+1})$ are identical for each k; a function $g_k(I_k, I_{k+1})$ may therefore be simply denoted by $g(I, I')$. Typical examples of decomposable causal independence models are the noisy-OR (Díez, 1993; Henrion, 1989; Pearl, 1988; Srinivas, 1993) and noisy-MAX (Díez, 1993; Heckerman and Breese, 1996; Srinivas, 1993) models, where the function g represents a logical OR and a MAX function, respectively.

In the case of continuous causal factors with a discrete effect variable, there are two main proposals for the conditional distribution of the discrete node (Bishop, 2006). Suppose we have a binary effect variable E and continuous parents $C_1, \ldots, C_n$. If E is modelled using a *logistic function*, then

$$P(e \mid C_1, \ldots, C_n) = \frac{\exp(b + w^T \varphi(C))}{1 + \exp(b + w^T \varphi(C))} \quad (2)$$

where $w^T = (w_1, \ldots, w_n)$ is a weight vector and $\varphi(C)$ a, possibly nonlinear, basis function applied to the causes C. The other option is to use the *probit regression model*, with

$$P(e \mid C_1, \ldots, C_n) = P(\Theta \leq (b + w^T \varphi(C))) \quad (3)$$

where $\Theta \sim N(0, 1)$. Although both types of model are flexible, it is very hard to come up with sensible weight vectors w and basis functions φ based only on available domain knowledge of the relations between causes.

3.3 The convolution theorem

A classical result from probability theory that is useful when studying sums of variables is the convolution theorem. The following well-known theorem (cf. (Grimmett and Stirzaker, 2001)) is central to the research reported in this paper.

Theorem 1. *Let f be a joint probability mass function of the random variables X and Y, such that $X + Y = z$. Then it holds that $P(X + Y = z) = f_{X+Y}(z) = \sum_x f(x, z - x)$.*

Proof. The (X, Y) space determined by $X + Y = z$ can be described as the union of disjoint sets (for each x): $\bigcup_x (\{X = x\} \cap \{Y = z - x\})$, from which the result follows. □

If X and Y are independent, then, in addition, the following corollary holds.

Corollary 1. *Let X and Y be two independent random variables, then it holds that*

$$\begin{aligned} P(X + Y = z) &= f_{X+Y}(z) \\ &= \sum_x f_X(x) f_Y(z - x) \quad (4) \end{aligned}$$

The probability mass function f_{X+Y} is in that case called the *convolution* of f_X and f_Y, and it is commonly denoted as $f_{X+Y} = f_X * f_Y$. The convolution theorem is very useful, as sums of random variables occur very frequently in probability theory and statistics. The convolution theorem can also be applied recursively, i.e.,

$$f_{X_1+\cdots+X_n} = f_{X_1} * \cdots * f_{X_n}$$

as follows from the recursive application of Equation (4):

$$P(X_1 + \cdots + X_n = z) = \sum_{y_{n-2}} \sum_{y_{n-3}} \cdots \sum_{y_1} \sum_{x_1} f_{X_1}(x_1) f_{X_2}(y_1 - x_1) \cdots f_{X_{n-1}}(y_{n-2} - y_{n-3}) f_{X_n}(z - y_{n-2}) \quad (5)$$

where we use the following equalities:

$$\begin{aligned} Y_1 &= X_1 + X_2 \\ Y_i &= Y_{i-1} + X_{i+1}, \ \ \forall i{:}\, 2 \leq i \leq n-2 \end{aligned}$$

Thus, $Y_{n-2} = X_1 + \cdots + X_{n-1}$, and $X_n = z - Y_{n-2}$. As addition is commutative and associative, any order in which the Y_i's are determined is valid.

The convolution theorem does not only hold for the addition of two random variables, but also for Boolean functions of random variables. However, in contrast to the field of real numbers where a value of a random variable X_n is uniquely determined by a real number z and y_{n-2} through $X_n = z - y_{n-2}$, in Boolean algebra values of Boolean variables only *constrain* the values of other Boolean variables. These constraints may yield a set of values, rather than a single value, which is still compatible with the convolution theorem. In the following, we use the notation $b(X, y) = z$ for such constraints, where the Boolean values y and z constrain X to particular values. For example, for $(X \vee y) = z$, where y, z stand for $Y = 1$ (Y has the value 'true') and $Z = 1$ (Z has the value 'true'), it holds that $X \in \{0, 1\}$.

Theorem 2. *Let f be a joint probability mass function of independent random, Boolean variables I and J and let b be a Boolean function defined on I and J, then it holds that*

$$P(b(I, J) = e) = \sum_i f_I(i) P(b(i, J) = e)$$

Proof. The (I, J) space defined by $b(I, J) = e$ can be decomposed as follows: $\bigcup_i \{I = i\} \cap \{J = j \mid b(i, j) = e\}$, where the expression $b(i, j) = e$ should be interpreted as a logical constraint on the Boolean values of the variable J. As in Theorem 1, the individual sets $\{I = i\} \cap \{J = j \mid b(i, j) = e\}$ are mutually exclusive. □

This theorem is illustrated by the following example.

Example 1. Consider the example given in Figure 1 as discussed in Section 2, and the Boolean relation $A \vee R \equiv L$, which expresses that fat loss L is due to changes in the energy balance A or fat removal R. By applying Theorem 2 the following results: $P(A \vee R = l) = \sum_a f_A(a) P(a \vee R = l) = f_A(a)\left(f_R(r) + f_R(\bar{r})\right) + f_A(\bar{a}) f_R(r) = f_A(a) f_R(r) + f_A(a) f_R(\bar{r}) + f_A(\bar{a}) f_R(r)$, where the term $(f_R(r) + f_R(\bar{r}))$ results from the logical constraint that $a \vee R = l$, i.e., $R \in \{0, 1\}$. Note that this is exactly the same result as for the noisy-OR model with the causal variables C marginalised out:

$$P_\vee(l) = \sum_{a \vee r = l} f_A(a) f_R(r) = P(A \vee R = l)$$

4 Convolution-based Causal Independence

In this section, we start to systematically explore the relationship between the convolution theorem of probability theory and the theory of causal independence.

4.1 General idea

The idea now is that we can use any Boolean-valued function, as long as the function is decomposable, to model causal interaction using the convolution theorem. A discrete causal independence model can also be written as follows:

$$P_b(e \mid C) = P(b(I_1, \ldots, I_n) = e \mid C)$$

where the right hand side can be determined as follows:

$$P(b(I_1, \ldots, I_n) = e \mid C) = \sum_{j_{n-2}} \sum_{j_{n-3}} \cdots \sum_{j_1} \sum_{i_1} f_{I_1}(i_1 \mid C_1) \cdot P_{I_2}(b_1(i_1, I_2) = j_1 \mid C_2) \cdots P_{I_n}(b_{n-1}(j_{n-1}, I_n) = e \mid C_n) \quad (6)$$

and the Boolean random variables J_k are defined in terms of I_l's dependent on the constraints imposed by the Boolean operators b_k. This can be proven by an inductive argument over all the cause variables. If we use a single operator $\odot$ that is commutative and associative, then the order of evaluation does not matter, and we can ignore parentheses: $b(I_1, \ldots, I_n) = I_1 \odot \cdots \odot I_n$ (Zhang and Poole, 1996; Lucas, 2005). However, if the single operator used to define the Boolean function b is neither commutative nor associative, then the order in which the Boolean expression is evaluated matters, and one should use parentheses.

The principles discussed above carry over to the continuous case. The convolution theorem for continuous variables X, Y, and Z, with $Z = X + Y$, has the following form:

$$f_{X+Y}(z) = \int_{-\infty}^{\infty} f_X(x) f_Y(z - x)\,\mathrm{d}x$$

where f_{X+Y}, f_X, and f_Y are probability density functions, and the variables X and Y are assumed to be independent. In the context of the theory of causal independence, we use convolution to compute the conditional probability density function $f_b(e \mid C)$, in a way very similar to the discrete case, where b is the causal interaction function.

4.2 A language for modelling interactions

To carry over the ideas of causal independence from the discrete case, we consider various operators for continuous variables. This will build up a rich language for modelling causal independence.

4.2.1 Boolean-valued continuous operators

Moving to the continuous case, first let I be a set of independent continuous causal random variables with associated probability density $f(I \mid C)$. Consider the Boolean-valued decomposable functions b, i.e., functions $b : I \to \{0, 1\}$, such that constraints on some variables $I' \subset I$ imposed by b are measurable sets of values for I'. We now wish to use the theory of causal independence in order to decompose the probability mass $f_b(e \mid C)$. If $I = \{J, K\}$ are continuous intermediate variables and $C = \{C_J, C_K\}$ the relevant causal variables, then:

$$\begin{aligned} f_b(e \mid C) &= P(b(J,K) = e \mid C) \\ &= \iint_{b(j,k)=e} f_{JK}(j,k \mid C)\,\mathrm{d}k\,\mathrm{d}j \\ &= \int_{-\infty}^{\infty} f_J(j \mid C_J) \int_{b(j,k)=e} f_K(k \mid C_K)\,\mathrm{d}k\,\mathrm{d}j \quad (7) \\ &= \int_{-\infty}^{\infty} f_J(j \mid C_J) P(b(j,K) = e \mid C_K)\mathrm{d}j \quad (8) \end{aligned}$$

The constraint $b(j, K) = e$ determines a subspace of the real numbers for variable K over which the density function f_K is integrated.

For a general n-ary Boolean-valued function b of continuous variables, we can apply this equation recursively, which gives:

$$\begin{aligned} f_b(e \mid C) = P(b(I_1, I_2, \ldots, I_n) = e \mid C) = \\ \int_{-\infty}^{\infty} f_{I_1}(i_1 \mid C_1) \int_{b(i_1,i_2,\ldots,i_n)=e} f_{I_2}(i_2 \mid C_2) \cdots \\ \cdot \int_{b(i_1,\ldots,i_n)=e} f_{I_n}(i_n \mid C_n)\,\mathrm{d}i_n \cdots \mathrm{d}i_1 \quad (9) \end{aligned}$$

If b is defined on both discrete and continuous variables, then this yields a mix of sums and integrals by repeated application of Theorem 2 and Eq. (8).

Analogously to the convolution notation, we define an operator $\textcircled{b}$ for denoting this decomposition for any Boolean function such that:

$$\textcircled{b}\,(f_{I_1}^{C_1}, \ldots, f_{I_n}^{C_n})(e) = f^{C}_{b(I_1,\ldots,I_n)}(e) = f_b(e \mid C)$$

where the superscripts C_1 and C_2 represent conditioning on the corresponding variables. This allows us to deal with complex combinations of such operators in a compact fashion.

If b is binary, we use an infix notation; e.g., $\textcircled{\vee}$ denotes the decomposition of two densities f_J and f_K using a logical OR. Returning to the fat loss problem (denoted by the variable L with l standing for $L = 1$) of Example 1, we have:

$$(f_A \mathbin{\textcircled{\vee}} f_R)(l) = \sum_a f_A(a) P((a \vee R) = l)$$

which is again the noisy-OR operator.

In the following section, a language that supports Boolean combinations of relations is developed.

4.2.2 Relational operators

The relational operators are treated similarly to convolutions and Boolean operators by viewing a

relation and a value of a random variable as a constraint on the other variables. First, basic operators to build up our language are basic relational operators, such as $=, \leq, >$. Consider $\leq$:

$$P_{\leq}(e \mid C) = P((I_1 \leq I_2) = e \mid C) = \iint_{(i_1 \leq i_2)=e} f(i_1, i_2 \mid C)\,\mathrm{d}i_1\,\mathrm{d}i_2 \quad (10)$$

If I_1 and I_2 independent, then the following equality results:

$$\begin{aligned} P_{\leq}(e \mid C) &= \int_{-\infty}^{\infty} f_{I_1}(i_1 \mid C_1) \cdot P((i_1 \leq I_2) = e \mid C_2)\,\mathrm{d}i_1 \\ &= \int_{-\infty}^{\infty} f_{I_1}(i_1 \mid C_1) \cdot \int_{i_i}^{\infty} f_{I_2}(i_2 \mid C_2)\,\mathrm{d}i_2\,\mathrm{d}i_1 \end{aligned}$$

A similar expression can be derived for $>$, while $P((I_1 = I_2) = e \mid C) = 0$ as $P((I_2 = i_1 \mid C_2) = 0$ for continuous variables I_1 and I_2. This expression implies that, in case I_1 and I_2 are independent, the relation can be decomposed. As a result, we can use the notation as introduced earlier to obtain operators $\circledR$:

$$\begin{aligned} (f_{I_1}^{C_1} \circledR f_{I_2}^{C_2})(e) &= f_R(e \mid C) \\ &= P(R(I_1, I_2) = e \mid C) \end{aligned}$$

where R is one of the basic relational operators.

Subsequently, we look at the extension of this language with convolutions of the interaction between variables and constants. A constant k can be described by a uniform probability distribution with density function

$$f_J(j) = \begin{cases} 1/\delta & \text{if } j \in (k - \delta/2, k + \delta/2] \\ 0 & \text{otherwise} \end{cases}$$

for $\delta \in \mathbb{R}^+$ very small, then

$$\begin{aligned} P((I \leq J) = e) &= (f_I \circledS f_k)(e) \\ &= \int_{-\infty}^{k} f_I(i)\,\mathrm{d}i = P(I \leq k) \end{aligned}$$

as one would expect. For convenience, we have written f_k for this density function f_J and will do so in the following.

For modelling the interaction between convolutions of variables, let I a set of continuous random variables and K a set of constants. Then, a *sum-relation* is a Boolean-valued function b such that

$$b(I) = R(\sum_{k=1}^{n} V_k, \sum_{l=1}^{m} W_l)$$

where $V \subseteq I \cup K$, $W \subseteq I \cup K$, and R is a relational operator.

If V and W do not overlap in variables except for the constants, the sums of V and W are independent. In that case, the relation can be decomposed by Eq. (9). So we have the following proposition.

Proposition 1. *The causal independence model of a sum-relation $R(\sum_{k=1}^{n} V_k, \sum_{l=1}^{m} W_l)$ with continuous interaction variables I can be written as:*

$$\begin{aligned} &P(R(\sum_{k=1}^{n} V_k, \sum_{l=1}^{m} W_l) = e) \\ &\quad = (f_{V_1+\cdots+V_n} \circledR f_{W_1+\cdots+W_m})(e) \end{aligned}$$

if $V \cap W \cap I = \varnothing$.

Example 2. Recall the example in Figure 1 as discussed in Section 2. The causal independence model of the energy balance A can be written as:

$$\begin{aligned} &P((I \leq H + W) = a \mid C, B, Y) \\ &= (f_I^C \circledS f_{H+W}^{\{B,Y\}})(a) = (f_I^C \circledS (f_H^B * f_W^Y))(a) \end{aligned}$$

where $*$ is the convolution operator.

This approach could be extended easily to other operators, such as subtraction, but we refrain from this because of space limitations.

4.2.3 Boolean combinations of relations

Sum-relations can now be combined using Boolean functions in a uniform manner. Let I_c be a set of continuous causal random variables, I_d a set of discrete causal random variables, and $I = I_c \cup I_d$. A *Boolean combination bc* is a Boolean-valued function defined on I as follows:

$$bc(I) = b(R_1(V_1), \ldots, R_n(V_n), I_d)$$

where b is a Boolean function and $R_1, \ldots, R_n$ a set of sum-relations.

If the continuous variables in the Boolean combinations of relations are partitioned, Eq. (6) can be applied to obtain the following proposition.

Proposition 2. *The causal independence model of a Boolean combination of sum-relations $b(R_1(V_1), \ldots, R_2(V_n))$, can be written as:*

$$P(b(R_1(V_1), R_2(V_2)) = e \mid C) = (f^{C_1}_{R_1(V_1)} \textcircled{b} f^{C_2}_{R_2(V_2)})(e)$$

if $V_1 \cap V_1 = \varnothing$.

Example 3. Again, consider the example in Figure 1 as discussed in Section 2. We are now in the position to decompose the full causal independence function representing fat loss L.

$$\begin{aligned} &P((I \leq H + W) \vee R) = l \mid C, B, Y, S) \\ &= P((R \vee (I \leq H + W)) = l \mid C, B, Y, S) \\ &= f^{\{C,B,Y,S\}}_{R \vee (I \leq H+W)}(l) \\ &= (f^S_R \textcircled{\vee} f_{\mathrm{L} \leq \mathrm{H+W}})(l) \\ &= (f^S_R \textcircled{\vee} (f^C_I \textcircled{\leq} (f^B_H * f^Y_W)))(l) \end{aligned}$$

5 Special Probability Distributions

In this section, the theory developed in the previous sections is illustrated by actually choosing special probability distributions to model problems.

5.1 Bernoulli distribution

As an example of discrete distributions, we take the simplest one: the Bernoulli distribution. This distribution has a probability mass function f such that $f(0) = 1 - p$ and $f(1) = p$. Let $P(I_k \mid c_k)$ be Bernoulli distributions with parameters p_k where $k = \{1, 2\}$. Suppose the interaction between C_1 and C_2 is modelled by $\leq$, then the effect variable E also follows a Bernoulli distribution with parameter:

$$\begin{aligned} P_{\leq}(e \mid c_1, c_2) &= (f^{c_1}_{I_1} \textcircled{\leq} f^{c_1}_{I_2})(e) \\ &= \sum_{i_1} f_{I_1}(i_1 \mid c_1) P((i_1 \leq I_2) = e \mid c_2) \\ &= p_1 - p_1 p_2 + 1 \end{aligned}$$

By the same reasoning, we obtain the parameters of the resulting distribution when $\bar{c}_1$ or $\bar{c}_2$.

5.2 Exponential distribution

In order to model the time it takes for the effect to take place due to the associated cause, we use the exponential probability distribution with distribution function $F(t) = 1 - e^{-\lambda t}$, where $t \in \mathbb{R}^+_0$ is the time it takes before the effect occurs. The associated probability density function is $f(t) = F'(t) = \lambda e^{-\lambda t}$. Now, let I_1 and I_2 stand for two of such temporal random variables such that $I_1 \leq I_2$, meaning that intermediate effect I_1 does not occur later than I_2. The probability mass of E to occur is:

$$\begin{aligned} P_{\leq}(e \mid C) &= (f^{c_1}_{I_1} \textcircled{\leq} f^{c_1}_{I_2})(e) \\ &= \int_{-\infty}^{\infty} f_{I_1}(i_1 \mid c_1) P((i_1 \leq I_2) = e \mid c_2) \,\mathrm{d}i_1 \\ &= \int_{-\infty}^{\infty} f_{I_1}(i_1 \mid c_1) \int_0^{\infty} f_{I_2}(i_1 + \delta \mid c_2) \,\mathrm{d}\delta \,\mathrm{d}i_1 \\ &= \int_{-\infty}^{\infty} \lambda_1 e^{-\lambda_1 i_1} e^{-\lambda_2 i_1} \,\mathrm{d}i_1 = \frac{\lambda_1}{\lambda_1 + \lambda_2} \end{aligned}$$

where we use a delay $\delta \geq 0$. If $\lambda_1 = \lambda_2$, then $P_{I_1 \leq I_2}(e \mid C) = 1/2$.

5.3 Conditional Gaussian distribution

The most common hybrid distribution for Bayesian networks is the conditional Gaussian distribution (Lauritzen and Wermuth, 1989). We illustrate the theory for the case when a continuous interaction variable I has a continuous cause variable C. The distribution of I is given in this model by $f(i \mid C) = N(\alpha + \beta C, \sigma^2)$. Let I_1 and I_2 be two such random variables with causal variables C_1 and C_2. It is well-known that variable E with $f_{I_1 - I_2}(e \mid C)$ is distributed Gaussian with mean $\alpha_1 + \beta_1 C_1 - \alpha_2 - \beta_2 C_2$ and variance $\sigma_1^2 + \sigma_2^2$. Similarly, the convolution of two Gaussian variables is a Gaussian variable with the sums of means and variances. Because of space limitations, the derivations are omitted.

Here we illustrate the relational operator $\leq$. The probability $P_{\leq}(e \mid C)$ can be obtained by

$$\begin{aligned} P_{\leq}(e \mid C) &= f^{C_1}_{I_1} \textcircled{\leq} f^{C_2}_{I_2} \\ &= (f^{C_1}_{I_1} \textcircled{-} f^{C_2}_{I_2}) \textcircled{\leq} 0 = F_J(0) \\ &= \tfrac{1}{2}\left[1 + \mathrm{erf}\left(\frac{-(\alpha_1 + \beta_1 c_1 - \alpha_2 - \beta_2 c_2)}{\sqrt{2(\sigma_1^2 + \sigma_2^2)}}\right)\right] \\ &= P(\Theta \leq b + w_1 c_1 + w_2 c_2) \end{aligned}$$

where $b = \frac{\alpha_2 - \alpha_1}{\sqrt{\sigma_1^2 + \sigma_2^2}}$, $w_1 = \frac{-\beta_1}{\sqrt{\sigma_1^2 + \sigma_2^2}}$, $w_2 = \frac{\beta_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}$, and $\Theta \sim N(0, 1)$, which is a probit regression model (cf. Section 3.2).

Example 4. Consider the energy balance A as decomposed in Example 2. Suppose all causal and interaction variables are conditionally Gaussian. Suppose the balance is negative, i.e., a is true, then,

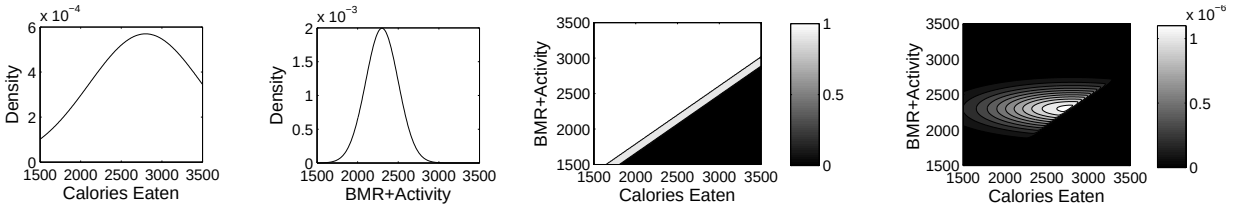


Figure 3: Example distributions, where, from left to right, the first figure shows the density of $C \sim N(2800, 700)$; the second figure shows the density of $B + Y \sim N(2300, 200)$; the third figure shows the probability distributions $P(A \mid C, B + Y)$ with $A \equiv I \leq (H + W)$ where $I \sim N(0.9 \cdot C, 200)$ and $H + W \sim N(1.1 \cdot (B + Y), 300)$; finally, the figure on the right shows the joint density of $\{A, C, B + Y\}$.

$(f_H^B * f_W^Y)(a)$ represents a distribution $N(\alpha_H + \alpha_W + \beta_H C_B + \beta_W C_Y, \sigma_H^2 + \sigma_W^2)$, i.e., the sum of the mean and variance. Using the above, it follows that the probability of a is:

$$P(a) = (f_I^C \circledS (f_H^B * f_W^Y))(a)$$

which is a probit regression model with $b = (\alpha_I - \alpha_H - \alpha_W)/\sigma'$, $w_C = \beta_I/\sigma'$, $w_B = -\beta_H/\sigma'$, and $w_Y = -\beta_W/\sigma'$, where $\sigma' = \sqrt{\sigma_I^2 + \sigma_H^2 + \sigma_W^2}$.

In Figure 3 a number of plots are given to illustrate this model for some realistic parameters. Note that the energy balance distributions depicted in the third figure are split up into 0 (too much intake), 1 (too much energy expenditure), and an uncertain band in the middle.

6 Conclusions

We presented a new algebraic framework for causal independence modelling of Bayesian networks that goes beyond what has been available so far. In contrast to other approaches, the framework supports the modelling of discrete as well as of continuous variables, either separately or mixed.

The design of the framework was inspired by the convolution theorem of probability theory, and it was shown that this theorem easily generalises to convolution with Boolean-valued functions. We also studied a number of important modelling operators. Contrary to regression models, we were thus able to model interactions between variables using knowledge at hand. Furthermore, the theory was illustrated by a number of typical probability distributions which one needs to use when actually building Bayesian network models for problems. Finally, although some of the results suggest that standard tools for solving the inference problem can be used, such as the probit model for the conditional Gaussian distribution, more research is required and such we intend to undertake in the near future.

References

C.M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

F.J. Díez. 1993. Parameter adjustment in Bayes networks: the generalized noisy OR-gate. In *UAI'93*, pages 99–105.

G. Grimmett and D. Stirzaker. 2001. *Probability and Random Processes*. Oxford University Press, Oxford.

D. Heckerman and J.S. Breese. 1996. Causal independence for probabilistic assessment and inference using Bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics*, 26(6):826–831.

M. Henrion. 1989. Some practical issues in constructing belief networks. In J.F. Lemmer and L.N. Kanal, editors, *Uncertainty in Artificial Intelligence*, pages 161–173, Amsterdam. Elsevier.

S.L. Lauritzen and N. Wermuth. 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57.

P.J.F. Lucas. 2005. Bayesian network modelling through qualitative patterns. *AI*, 163:233–263.

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

S. Srinivas. 1993. A generalization of the noisy-OR model. In *UAI'93*, pages 208–215.

N.L. Zhang and D. Poole. 1996. Exploiting causal independence in Bayesian network inference. *JAIR*, 5:301–328.

Contextual Variable Elimination with Overlapping Contexts

Wannes Meert, Jan Struyf and Hendrik Blockeel
Katholieke Universiteit Leuven, Department of Computer Science, Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract

Belief networks (BNs) extracted from statistical relational learning formalisms often include variables with conditional probability distributions (CPDs) that exhibit a local structure (e.g, decision trees and noisy-or). In such cases, naively representing CPDs as tables and using a general purpose inference algorithm such as variable elimination (VE) results in redundant computation. Contextual variable elimination (CVE) partly addresses this problem by representing the BN in terms of smaller units called confactors. This leads to a more compact representation and faster inference. CVE requires that a variable's confactors are mutually-exclusive and exhaustive. We propose CVE-OC (CVE with overlapping contexts), which lifts these restrictions. This seemingly simple step shows to be powerful and allows for a more efficient encoding of confactors and a reduction of the computational cost. Experiments show that CVE-OC outperforms CVE on multiple problems.

1 Introduction

Belief networks (BNs) (Pearl, 1988) are a well-known formalism to represent probabilistic relations between variables. Their main advantage is that they allow one to graphically represent which variables are (in)dependent of which other variables. A BN takes the form of a directed acyclic graph in which the variables are nodes and the dependence information is encoded by means of the edges. The BN defines the variables' joint probability distribution, which is compactly represented as a product of factors. This product includes for each variable one factor, which is typically encoded as a table that represents the conditional probability distribution (CPD) of the variable given its parents in the BN. BN inference (computing the conditional probability of a query variable given certain evidence) can be performed by summing out all non-query non-evidence variables from the factorization. This is essentially what the variable elimination (VE) algorithm (Zhang and Poole, 1996) does.

A field where BNs are applied is *statistical relational learning* (SRL) (Getoor and Taskar, 2007). This is a research area that combines the elegant handling of uncertainty from probability theory with the capability of representing complex relational domains of first-order logic. A large number of the formalisms used in SRL can be converted into BNs (e.g., CP-logic, ProbLog, ICL, PRISM, BLP). For these formalisms, only exploiting the notion of independence does not yield the most efficient representation possible. The CPDs resulting from the conversion exhibit a particular internal structure, which we will call *local structure*. For example, a CPD may be given as a decision tree (Ramon et al., 2008), may express that the different conditions influence the variable independently (noisy-or or noisy-max), or may impose constraints on the range of a variable in a certain context (Meert et al., 2008). In these cases, a table based representation contains redundancies; an alternative representation that avoids these redundancies may yield more efficient inference.

Several methods have been proposed to exploit local structure. For instance, contextual variable elimination (CVE) (Poole and Zhang, 2003) uses a more compact representation for decision trees and more complex local structures that exhibit contextual independence. This reduces the tree-width of the network, thus al-

Pp. 193–201 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

lowing for more efficient inference. CVE is a generalization of probability trees and a comparison is made in (Poole and Zhang, 2003). Another example is multiplicative factorization (MF) (Díez and Galán, 2003) that can exploit local structures like independent causation (e.g. noisy-or/and). CVE and MF each exploit one particular type of structure, but cannot handle the other. CVE relies on contexts being mutually-exclusive and exhaustive (we will call this the MEE-restriction), which makes it unsuitable to combine it with MF.

There are also methods that utilize a preprocessing phase to compile the belief network into a different structure, which is optimized for answering multiple queries and allows efficient inference with particular types of local structure. Some known methods are the Arithmetic Circuits (AC) of (Chavira and Darwiche, 2007) and the AND/OR-trees of (Mateescu and Dechter, 2008). In SRL, for each query a new network is built (every time requiring a compilation), therefore, in this paper, we focus on the compilation-free methods.

The main contribution of this paper is that we show how in CVE the MEE-restriction can be lifted. This leads to a new method, *CVE-OC*: CVE with overlapping constraints. Lifting the MEE-restriction has two important consequences: (a) contexts can be encoded more compactly, with increased efficiency as a result, and (b) factorizations like MF can also be handled, which means that CVE-OC can exploit all structures that CVE and MF can exploit. An additional contribution, is that CVE-OC handles constraints on the range of multi-valued variables. This is an extension of CVE for which no concrete solution has been presented up till now.

This paper is organized as follows. We start by providing background on CVE and MF. After this we presents CVE-OC, the paper's main contribution. Next, we discuss its usefulness in the context of statistical relational learning. We present an experimental evaluation in that context, and finally present our conclusions.

2 Preliminaries

2.1 Contextual variable elimination (CVE)

CVE makes use of a more specific form of conditional independence known as *contextual independence* (Boutilier et al., 1996; Poole and Zhang, 2003).

Definition 1 (Contextual Independence)**.** Assume that $\mathbf{X}$ $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{C}$ are sets of variables. $\mathbf{X}$ and $\mathbf{Y}$ are contextually independent given $\mathbf{Z}$ and context $\mathbf{C} = \mathbf{c}$, with $\mathbf{c} \in \text{dom}(\mathbf{C})$, iff

$$Pr(\mathbf{X}|\mathbf{Y} = \mathbf{y} \wedge \mathbf{Z} = \mathbf{z} \wedge \mathbf{C} = \mathbf{c}) = Pr(\mathbf{X}|\mathbf{Z} = \mathbf{z} \wedge \mathbf{C} = \mathbf{c})$$

for all $\mathbf{y} \in \text{dom}(\mathbf{Y})$ and $\mathbf{z} \in \text{dom}(\mathbf{Z})$ such that $Pr(\mathbf{Y} = \mathbf{y} \wedge \mathbf{Z} = \mathbf{z} \wedge \mathbf{C} = \mathbf{c}) > 0$. We also say that $\mathbf{X}$ is *contextually independent* of $\mathbf{Y}$ given $\mathbf{Z}$ and context $\mathbf{C} = \mathbf{c}$ (if we drop $\mathbf{C} = \mathbf{c}$, we say $\mathbf{X}$ is conditionally independent from $\mathbf{Y}$ given $\mathbf{Z}$).

VE (Zhang and Poole, 1996) represents the joint distribution as a product of factors, in which each factor is a conditional probability table. CVE (Poole and Zhang, 2003) factorizes the joint distribution further by replacing each factor by a set of *contextual factors* or *confactors*. A confactor r_i consists of two parts: a *context* and a *table*:

$$\langle \underbrace{V_1 = v_{1,i} \wedge \ldots \wedge V_{k-1} = v_{k-1,i}}_{\text{context}} , \underbrace{factor_i(V_k, \ldots, V_m)}_{\text{table}} \rangle$$

The context is a conjunction of variable-value tests ($V_j = v_{j,i}$), which indicates the condition under which the table is applicable (if the context is "*true*" the table is always applicable). The context is used to split up factors into confactors based on Def. 1. The table stores probabilities for all value assignments of a set of zero or more variables $(V_k, \ldots, V_m)$.

The set of confactors that together represent the CPD of a variable V (the confactors *for* V) is mutually-exclusive and exhaustive (MEE). This means that for each possible value assignment for a variable V and its parents $\mathbf{pa}(V)$,

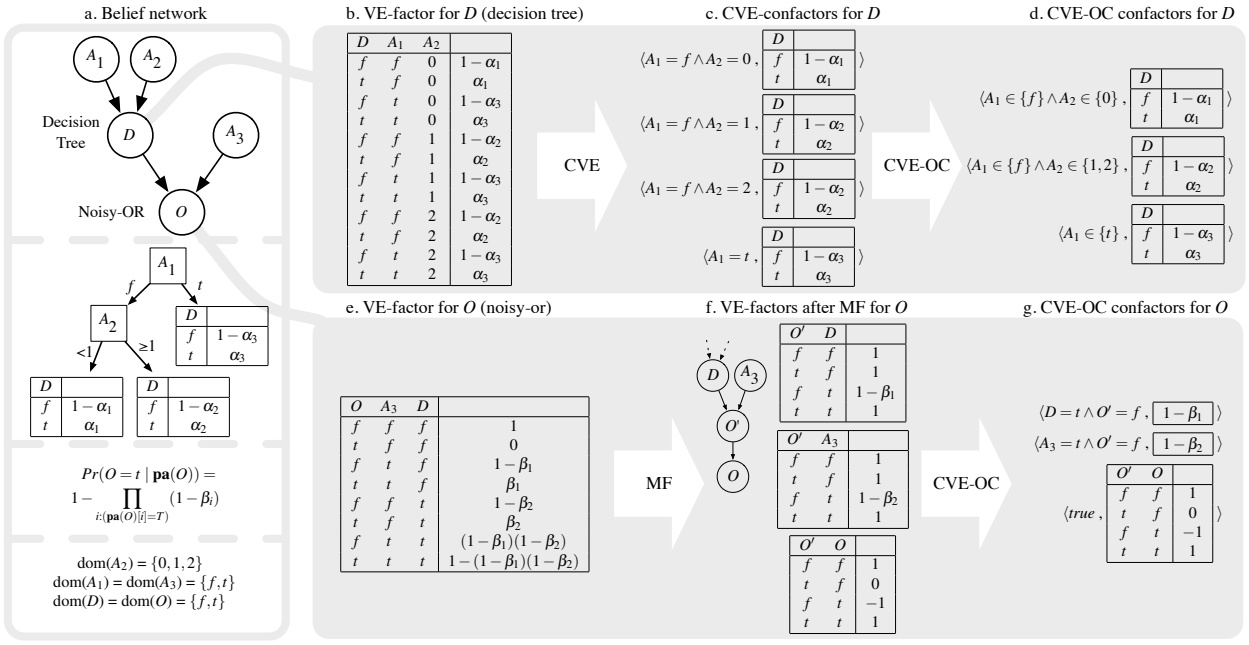


Figure 1: (a) Belief network with local structure; on top the graphical model, then the decision tree for node D and the definition of noisy-or for node O, and at the bottom the domains for the variables; (b) CPD for D represented as a table like in VE; (c) CPD for D represented by confactors for use in CVE; (d) CPD for D represented by confactors for CVE-OC; (e) CPD for O; (f) CPD for O factorized with MF for use in VE; (g) CPD for noisy-or represented by confactors for CVE-OC after MF.

there is precisely one confactor of which the table includes the parameter $Pr(V = v \mid \mathbf{pa}(V) = \mathbf{v_{pa}})$. These conditions ensure that confactors for the same variable do not overlap, therefore, the set of all confactors for a variable is identical to the original factor.

Fig. 1.c shows a confactor representation of a BN for which the CPD for D can be represented by a decision tree. This CPD can be compactly represented as a set of confactors of which each context is the conjunction of variable-value tests on a path from the decision tree root to one of its leaves. (As a note, the converse is not true; a decision tree cannot always represent confactors equally compactly.)

We describe CVE at a high level (the complete algorithm can be found in (Poole and Zhang, 2003)). Similar to VE, CVE eliminates the non-query, non-evidence variables one by one from the joint distribution. To eliminate a variable E, it relies on four basic operations:

1. $\langle c, t_1\rangle \otimes \langle c, t_2\rangle \equiv \langle c, t_1 \otimes t_2\rangle$, multiplying two confactors with identical contexts c.
2. $\sum_E \langle c, t\rangle \equiv \langle c, \sum_E t\rangle$, summing out a variable E that appears in the table of a confactor.
3. $\sum_E (\langle c \wedge E = e_1, t_1\rangle, \ldots, \langle c \wedge E = e_k, t_k\rangle) \equiv \langle c, \sum t_i\rangle$, summing out a variable E, with domain $e_1, \ldots, e_k$, that appears in the contexts.
4. $\langle c, t\rangle \equiv \langle c \wedge X = x_1, t(X = x_1)\rangle, \ldots, \langle c \wedge X = x_k, t(X = x_k)\rangle$, *splitting* a factor; $t(X = x_i)$ is table t but elements for which $X \neq x_i$ are removed.

The first three operations are only possible if the contexts are identical (indicated with c) except for the variable to eliminate (E). To make the contexts identical, CVE uses the fourth operator (splitting). Given two confactors, repeated splitting can be used to create two confactors with identical contexts. The order in

which these operators are applied is chosen by the so-called *absorption* algorithm for CVE. Splitting creates extra confactors, and therefore heuristics are used to avoid this operation as much as possible.

Confactors can represent CPDs more compactly than tables, but, as the previous discussion illustrates, at the cost of more complicated basic operations.

2.2 Multiplicative factorization of noisy-max

Fig. 1.e shows how noisy-or can be represented in terms of a table that is used by VE. This representation has the disadvantage that, while the inputs independently cause the output, this independence is not reflected in the factorization. Confactors do not offer a solution for this type of local structure, so another technique should be used.

(Díez and Galán, 2003) propose a state-of-the-art multiplicative[1] factorization (MF) for a factor representing noisy-or (and its generalization noisy-max). In this new set of factors, each factor only involves one of the inputs. In general, this leads to faster inference with VE. It is not necessary for this paper to fully understand the method or the example in Fig. 1.f, but important to note is that this method uses multiple factors to represent the CPD for a variable (in this case O'). Because of the MEE-restriction, these two factors cannot be represented by confactors.

3 CVE with overlapping contexts

Our main contribution is the CVE-OC algorithm. The CVE-OC algorithm removes the restrictions on the confactors imposed by the CVE algorithm:

First, CVE expects that the confactors for a variable V are MEE. This condition ensures that the parameters in the confactors are identical to those in V's original conditional probability table. It is also a pre-condition for the algorithm CVE uses to combine confactors while eliminating a variable (the absorption algorithm). As mentioned in the introduction, this pre-condition has certain disadvantages (e.g., it is incompatible with MF (Díez and Galán, 2003), and it may make expressing logical constraints more complicated). Therefore, we choose to remove this pre-condition. As a result, a parameter in the original table is not guaranteed to be equal to a parameter in a single confactor (like in CVE) but is equal to the multiplication of different parameters found in different confactors with non-mutually-exclusive contexts. As a consequence the absorption algorithm can no longer be used and we need a new technique to decide which confactors to combine when.

Second, the equality tests in the contexts can be replaced with set membership tests. This allows for a more compact representation in domains with multi-valued variables. This representation was already proposed by Poole and Zhang (Poole and Zhang, 2003), but not supported in their algorithm and implementation as it requires one to extend the splitting operation.

A confactor r_i now has the following form:

$$\langle V_1 \in v_{1,i} \wedge \ldots \wedge V_k \in v_{k,i} \wedge \ldots \wedge V_n \in v_{n,i} \quad , \\ factor_i(V_k, \ldots, V_n, \ldots, V_m)\rangle$$

The context is a conjunction of set membership tests ($V_j \in v_{j,i}$, $v_{j,i} \subseteq dom(V_j)$, with $1 \leq j \leq n$), which indicates the condition under which the table is applicable. The table stores probabilities for given value assignments for a set of zero or more variables ($V_k \ldots V_m$). Note that a variable can now appear both in the context and in the table.

The interpretation of a set of confactors with overlapping contexts for a variable V can be given in terms of the multiplication of their parameters. Given a value assignment for a variable and its parents, it is now possible that multiple confactors for V have contexts that are a applicable and each of these confactors has a parameter in its table that is consistent with the value assignment. The product of these parameters is equal to the parameter in the original

[1] The term 'multiplicative' is used because the summation that is typical for noisy-or is transformed into a multiplication causing further factorization.

table representing the CPD. If the set is not exhaustive, we assume that the value assignments not covered by a context correspond to parameters that are equal to 1.0, which are irrelevant in a multiplication.

Based on the above modifications, we can convert the CPDs in Fig. 1.c and 1.f into the more compact representation in Fig. 1.d and 1.g, which is the input to CVE-OC. This shows three new uses of confactors: (a) it is possible to use set membership to express value ranges of a variable (e.g., the conditions on A_2); (b) noisy-or can be represented more efficiently by using MF (Díez and Galán, 2003) (possibly combined with set membership in case of multi-valued variables); (c) logical constraints can be more compactly expressed and will be exploited during variable elimination.

3.1 The CVE-OC algorithm

Recall from the explanation of CVE that its core operations are not only multiplication and sum-out, but also *compatibility checking* and *splitting*. As explained in (Poole and Zhang, 2003), compatibility checking and splitting must be performed very often and may therefore be computationally expensive.

Since CVE-OC allows overlapping contexts, it cannot use *absorption* to reduce the number of compatibility checks. Moreover, the use of set membership tests requires even more types of splitting. To improve the efficiency of compatibility checking and splitting we propose a temporary tree-based index structure to represent the set R_E of all confactors that contain the variable E that is being eliminated.

Fig. 2 shows an example of this index structure. Each internal node contains a variable that appears in a context of a confactor in R_E and the outgoing edges of the node are labeled with subsets of the variable's domain. The leaf nodes contain the tables.

Before explaining the index construction procedure, we define the *rank* of a variable. Each variable is given a different rank. $rank(E) = 0$, and the ranks of the other variables are positive and increase monotonically with the number of confactors they appear in (ties are broken at random). The index will have the property that variables with a higher rank occur higher up in the tree.

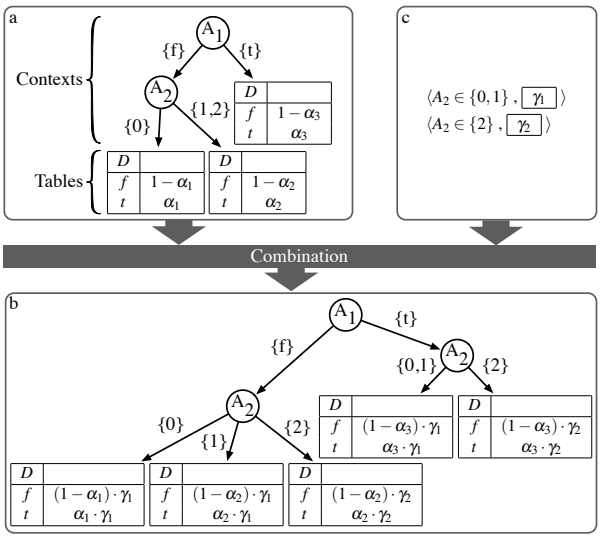


Figure 2: (a) Tree index structure used by CVE-OC to eliminate A_2, after adding the confactors from Fig. 1.d. (b) The tree structure after adding the confactors shown in (c) to (a).

To construct the index, CVE-OC starts with a tree that consists of a single leaf that contains a table $t = 1.0$. Then it absorbs all confactors from R_E one by one into the tree. In the end, the tree will contain for any context a correct CPD. To absorb a confactor $r = \langle c, t \rangle$, it moves the confactor down the tree. When it encounters an internal node containing variable N, one of the following five cases may occur (CVE-OC acts according to the first case that applies).

1. c's top-ranked variable V has a higher rank than rank(N), i.e., V should appear above N in the tree. V occurs as $V \in v$ in c. CVE-OC then replaces node N by a new node labeled V with two outgoing edges: one labeled v, and one labeled with its complement $(\mathrm{dom}(V) - v)$. The original subtree rooted at N is duplicated and becomes both the left and right subtree of the new node V. CVE-OC removes variable V from r's context c and sorts the resulting confactor down both subtrees.

2. N appears in r's context, i.e., $(N \in n) \in c$. If one of the outgoing edges from N is labeled n, then CVE-OC sorts r down that branch. If not, then it must perform the splitting operation. To this end, it processes all N's outgoing edges in turn. For a given edge i, if its label n_i is disjoint from n, the corresponding subtree is incompatible with r and no further computation is required. Otherwise, n_i must be split. CVE-OC removes the edge labeled n_i from N and replaces it with two new edges, one labeled $n_i^+ = n_i \cap n$ and one labeled $n_i^- = n_i - n$. The original subtree rooted at n_i is duplicated below these two new edges. Next, CVE-OC removes N from r's context and projects its table on the condition $N \in n_i^+$. Then it sorts the resulting confactor down edge n_i^+.

3. N appears in r's table. Let $n_1, \ldots, n_k$ be the labels of N's outgoing edges. CVE-OC sorts r down the tree via each n_i after projecting r's table on the condition $N \in n_i$.

4. r represents a logical constraint $(t = 0)$ and $c = \emptyset$. In this case, CVE-OC replaces node N by a new leaf and initializes the table in that leaf to zero. We call this step pruning the tree based on a constraint.

5. CVE-OC sorts r down to all subtrees of N.

If r reaches a leaf, which stores a table t_l, then the following happens. If $t_l = 0$, then this leaf represents a constraint and no further computation is required. If $t_l \neq 0$, then there are two cases: either $c = \emptyset$ or $c \neq \emptyset$. In the former case, CVE-OC replaces the table in the leaf by the product of t_l and t. In the latter case, it must introduce a new node for the top ranked variable in c into the tree. This is done in precisely the same way as in step one above.

After all confactors in R_E are absorbed into the index, CVE-OC can sum-out the variable E. This is done in two steps. In the first step, E is summed out from all the tables in the leaves of the index. The next step applies to all internal nodes that are labeled with E. Because $\text{rank}(E) = 0$, all children of such nodes are leaves. To sum-out E, a node that contains E is simply replaced by a leaf with a table equal to the sum of the tables in E's children.

After E is eliminated, the tree is converted back into a set of confactors, by creating one confactor for every leaf. Together with the confactors not in R_E, these form the set of confactors for the following elimination step. Note that we cannot reuse the same index for eliminating a different variable because the index is specific to the variable that is being eliminated. Also, converting all confactors into one single tree is to be avoided as a tree is in general not the most compact representation for a set of confactors. Therefore, we include as few as possible confactors in the tree (by means of the variable ordering), and we only use the tree to make compatibility checking and splitting more straightforward.

4 Experiments

We evaluate the inference methods on the task of inferring the marginal distribution of one designated variable in four types of BNs of varying complexity. We always select the variable with the highest inference cost and do not include any evidence (i.e., we consider the most difficult case). The software and BNs are available online (dtai.cs.kuleuven.be/corporal). We compare five algorithm/input combinations: VE, VE with multiplicative factorization (VE+MF), CVE$^{\in}$, CVE-OC, and CVE-OC with multiplicative factorization (CVE-OC+MF). CVE$^{\in}$ is our own C++ implementation of CVE; it is the second algorithm described in (Poole and Zhang, 2003) (which uses absorption), extended with set membership tests (so we can accurately assess the contribution of the overlapping confactors). CVE-OC uses exactly the same input (confactors) as CVE$^{\in}$; CVE-OC+MF has additional confactors for noisy-or/and structures. We use the minimum deficiency elimination ordering (Bertele and Brioschi, 1972), which is a simple greedy heuristic that performs well for VE. Fig. 3 presents the results. Additionally, we have added in the graphs results obtained using the ACE system (Chavira and Darwiche,

2007), which is a representative and state-of-the-art compilation-based approach. Version 2.0 is used, with default settings. The input consists of conditional probability tables except for noisy-or/and nodes which are encoded using the noisy-max syntax.

The BNs in (a) and (b) are constructed from artificial CP-logic theories (Vennekens et al., 2009). The BNs in (a) only include interconnected noisy-or (white) and and (black) nodes with a linearly increasing number of parents. VE runs out of memory when the the number of noisy-or nodes is larger than 8, while CVE can handle larger BNs because of its compact confactor representation. VE+MF and CVE-OC+MF are much faster than the other methods because they efficiently factorize the noisy-or nodes. For experimental comparison of MF with other methods for noisy-or networks we refer to (Díez and Galán, 2003; Savicky and Vomlel, 2007). The ACE system also exploits the presence of noisy-or/and nodes but is a factor 100 slower because of the compilation where it optimizes for all variables. After crossing the curves for non-MF methods, the ACE-curve stops because the tables used as input become too large to generate. The network in (b) contains many CPDs structured as decision trees (black) with a linearly increasing number of parents. CVE and CVE-OC are well suited to handle such a representation and are therefore faster than VE (MF has no influence). The ACE system seems to be unable to fully exploit the interconnected decision tree structures in the CPDs and is also slower because it tries to optimize the structure for all variables.

The BNs in (c) are constructed from a CP-theory that was learned from the UW-CSE dataset (Richardson and Domingos, 2006). The UW-CSE BNs include all structures also included in (a) and (b). For such networks CVE-OC+MF excels as it can efficiently represent all these structures. The other methods run out of memory because they cannot represent one of the local structures efficiently. The noisy-or/and nodes are no problem for the ACE system and when the decision tree structures do not interact too heavily it can handle them efficiently. The curve for CVE-OC+MF fluctuates because the heuristic used for the elimination ordering is too agnostic about local structure when combining different structures. A better heuristic is an interesting future research topic.

The networks in (d) are the randomized networks used in Fig. 11 of (Poole and Zhang, 2003) and created with the original Java code available from the author. This experiment compares CVE$^{\in}$ with the new CVE-OC algorithm on the same data as Poole and Zhang used. This shows that CVE-OC, although more generally applicable, is about as fast as CVE$^{\in}$ even when there are no additional structures to be exploited.

5 Conclusions and future work

We presented the algorithm CVE-OC (CVE with overlapping contexts), which extends contextual variable elimination (CVE). The introduction of overlapping contexts is a simple but powerful step. From the representation point of view, it offers an elegant combination of deterministic and probabilistic knowledge. From the computational point of view, the need for equality testing is reduced, invalid combinations of values are pruned, and the loosened restrictions allow for better optimizations. CVE-OC generalizes over both CVE and MF, and provide optimization opportunities beyond the union of what those methods offer.

The experiments show that CVE-OC, while more generally applicable, can handle input for CVE without any loss in efficiency. Because of its generality, the input for CVE-OC can be more compact than for CVE and other known optimization methods for VE can be integrated. For example, the integration of MF is shown to be faster and more compact than only VE, CVE or MF.

In future work, we intend to perform more experiments on the propagation of constraints in CVE-OC and would like to investigate more the influence of heuristics for elimination orderings. With respect to SRL, we would like to investigate further the relationship between CVE-OC and other SRL inference algorithms and see how

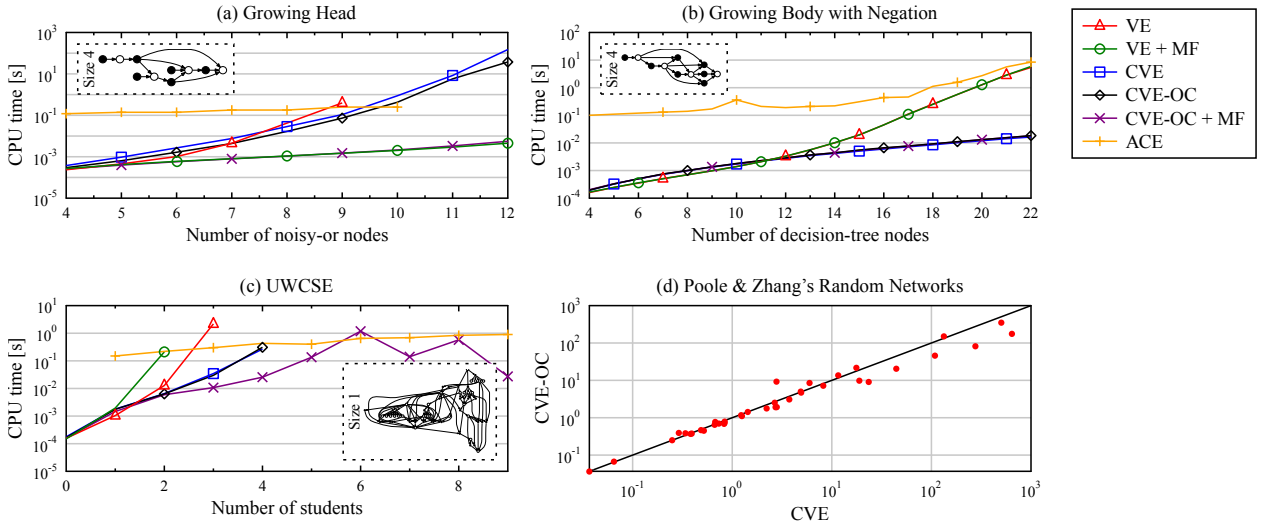


Figure 3: Inference times for networks originating from CP-theories and random networks. The horizontal axis of (a)-(c) indicates BN complexity.

we can integrate this into learning methods for SRL.

Acknowledgements

Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) to Wannes Meert. GOA/08/008 'Probabilistic Logic Learning'.

References

U. Bertele and F. Brioschi. 1972. *Nonserial Dynamic Programming*. Academic Press.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. 1996. Context-specific independence in Bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 115–123.

M. Chavira and A. Darwiche. 2007. Compiling Bayesian networks using variable elimination. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2443–2449.

F.J. Díez and S.F. Galán. 2003. Efficient computation for the noisy MAX. *Intelligent Systems*, 18(2):165–177.

L. Getoor and B. Taskar, editors. 2007. *Statistical Relational Learning*. MIT Press.

R. Mateescu and R. Dechter. 2008. Mixed deterministic and probabilistic networks. ICS technical report, University of California, Irvine, May.

W. Meert, J. Struyf, and H. Blockeel. 2008. Learning ground CP-logic theories by leveraging Bayesian network learning techniques. *Fundamenta Informaticae*, 89(1):131–160.

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

D. Poole and N. Zhang. 2003. Exploiting contextual independence in probabilistic inference. *Artificial Intelligence Research*, 18:263–313.

J. Ramon, T. Croonenborghs, D. Fierens, H. Blockeel, and M. Bruynooghe. 2008. Generalized ordering-search for learning directed probabilistic logical models. *Machine Learning*, 70(2-3):169–188.

M. Richardson and P. Domingos. 2006. UW-CSE. http://alchemy.cs.washington.edu/data/uw-cse/.

P. Savicky and J. Vomlel. 2007. Exploiting tensor rank-one decomposition in probabilistic inference. *Kybernetika*, 43(5):747–764.

J. Vennekens, M. Denecker, and M. Bruynooghe. 2009. CP-logic: A language of causal probabilistic events and its relation to logic programming. *Theory and Practice of Logic Programming*, 9(3):245–308.

N. Zhang and D. Poole. 1996. Exploiting causal independence in bayesian network inference. *Artificial Intelligence Research*, 5:301–328.

Honour Thy Neighbour—Clique Maintenance in Dynamic Graphs

Thorsten J. Ottosen
Department of Computer Science, Aalborg University, Denmark
nesotto@cs.aau.dk

Jiří Vomlel
Institute of Information Theory and Automation of the AS CR, The Czech Republic
vomlel@utia.cas.cz

Abstract

Whenever objects and their interaction is modelled via undirected graphs, it is often of great interest to know the cliques of the graph. For several problems the graph changes frequently over time, and we therefore seek methods for updating the information about the cliques in a dynamic fashion to avoid expensive recomputations. This dynamic problem was investigated by Stix, and in this paper we derive a new simple method based on the Bron-Kerbosch algorithm that compares favourably to Stix' approach. The new approach is generic in the sense that it can be used with other algorithms than just Bron-Kerbosch. The applications include fuzzy clustering and optimal triangulation of Bayesian networks.

1 Introduction

We consider the problem of maintaining the set of cliques of a dynamic undirected graph. The graph is dynamic in the sense that edges can be removed and added, but the set of vertices is invariant. When we add a new set of edges, we call the problem *incremental*, and when we remove a set of edges, we call the problem *decremental*. Finding all the cliques of a static graph is a hard problem: the clique decision problem is NP-complete (Karp, 1972) and listing all the cliques may require exponential time as there exists graphs with exponentially many cliques (Moon and Moser, 1965)—albeit it is solvable in polynomial time for many classes of graphs. However, in this work we shall consider the initial set of cliques for given (several well-known algorithms exists for this purpose).

Previous research has been motivated by fuzzy clustering (Stix, 2004), but we have another application in mind. Specifically, our motivation is to find optimal triangulations of Bayesian networks with respect to the total table size by using a best-first or depth-first search. This requires a lower bound on the total table size for which we may use the total table size of the current partially triangulated graph. In turn, this requires that we know the cliques of the current graph. A detailed description of the new best-first and depth-first approach to triangulation can be found in (Ottosen and Vomlel, 2010).

We shall use the following notation and definitions. $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ is an *undirected graph* with *vertices* $\mathrm{V} = \mathcal{V}(\mathrm{G})$ and *edges* $\mathrm{E} = \mathcal{E}(\mathrm{G})$. For a set of edges F, $\mathcal{V}(\mathrm{F})$ is the set of vertices $\{\mathrm{u}, \mathrm{v} : \{\mathrm{u}, \mathrm{v}\} \in \mathrm{F}\}$. For $\mathrm{W} \subseteq \mathrm{V}$, $\mathrm{G}[\mathrm{W}]$ is the *subgraph induced by* W. Two vertices u and v are *connected* in G if there is an edge between them. A graph G is *complete* if all pairs of vertices $\{\mathrm{u}, \mathrm{v}\}$ $(\mathrm{u} \neq \mathrm{v})$ are connected in G. A set of vertices $\mathrm{W} \subseteq \mathrm{V}$ is *complete* in G if $\mathrm{G}[\mathrm{W}]$ is a complete graph. If W is complete and no complete set U exists such that $\mathrm{W} \subset \mathrm{U}$, then W is a *clique*. (Remark: note that any complete set is sometimes called a clique; then what we defined as a clique is called a maximal clique.) The *set of all cliques* of a graph is denoted $\mathcal{C}(\mathrm{G})$ and the set of all cliques that intersects with a set of vertices W is denoted $\mathcal{C}(\mathrm{W}, \mathrm{G})$. For a single vertex v we also allow the notation

Pp. 201–209 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

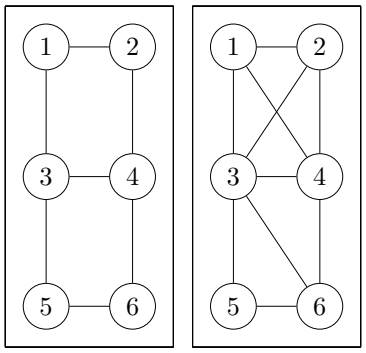


Figure 1: Left: The initial graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$. Right: The updated graph G'. We have $\mathcal{C}(\mathrm{G}) = \mathrm{E}$ and $\mathcal{C}(\mathrm{G}') = \{\{1,2,3,4\},\{3,5,6\},\{3,4,6\}\}$. So in this example we have $\mathcal{RC}(\mathrm{G},\mathrm{G}') = \mathcal{C}(\mathrm{G})$ and $\mathcal{NC}(\mathrm{G},\mathrm{G}') = \mathcal{C}(\mathrm{G}')$.

$\mathcal{C}(\mathrm{v},\mathrm{G})$. The *neighbours* of a set of vertices W are those vertices from $\mathrm{V}\setminus\mathrm{W}$ that are connected to at least one vertex $\mathrm{v} \in \mathrm{W}$ and we write this as $\mathrm{nb}(\mathrm{W},\mathrm{G})$. Similarly, $\mathrm{fa}(\mathrm{W},\mathrm{G})$ is the *family* of W in G, that is, the set $\mathrm{nb}(\mathrm{W},\mathrm{G}) \cup \{\mathrm{W}\}$. As usual we allow the notation $\mathrm{nb}(\mathrm{v},\mathrm{G})$ and $\mathrm{fa}(\mathrm{v},\mathrm{G})$. A vertex v is *simplicial* if $\mathrm{nb}(\mathrm{v},\mathrm{G})$ is complete. If $\mathrm{G}' = (\mathrm{V}, \mathrm{E} \cup \mathrm{F})$ is the graph resulting from adding a set of new edges F to G, then $\mathcal{RC}(\mathrm{G},\mathrm{G}') = \mathcal{C}(\mathrm{G}) \setminus \mathcal{C}(\mathrm{G}')$ is the set of *removed cliques*, and $\mathcal{NC}(\mathrm{G},\mathrm{G}') = \mathcal{C}(\mathrm{G}') \setminus \mathcal{C}(\mathrm{G})$ is the set of *new cliques*. Figure 1 illustrates these concepts. Finally, a complete set of vertices C in G' is called a *clique candidate* for G'.

2 Stix' Approach To Clique Maintenance

Stix observed that it was somewhat expensive to recompute all cliques of a graph given that the graph had only changed slightly. Therefore Stix derived the approach explained below and showed that it did indeed out-perform a full recomputation scheme (Stix, 2004).

Stix' approach works by adding (removing) one edge at a time. To add (remove) a set of edges, the technique is simply applied once for each edge. The technique (both for incremental and decremental problems) may be summarized as follows: (1) Let $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ be an undirected graph, and let $\mathrm{G}' = (\mathrm{V}, \mathrm{E} \cup \{\{\mathrm{v},\mathrm{w}\}\})$. (2) Initially let $\mathcal{C} = \mathcal{C}(\mathrm{G})$. (3) Generate a set of clique candidates $\mathcal{K}$ for the updated graph G'. (4) Add/remove a candidate $\mathrm{C} \in \mathcal{K}$ to/from $\mathcal{C}$ depending on whether it is a clique in G'. (5) In the end, $\mathcal{C}$ equals $\mathcal{C}(\mathrm{G}')$

The check in step 4 is shown in Algorithm 1 where we have improved Stix' approach by only considering the neighbours $\mathrm{nb}(\mathrm{C},\mathrm{G}')$ of a clique candidate C (notice that this algorithm should be called with the updated graph G' as its second argument).

Stix' algorithm is based on the following theorem:

Theorem 1. *(Stix, 2004) Let* $\mathrm{G} = (\mathrm{V},\mathrm{E})$ *be an undirected graph, and let* $\mathrm{G}' = (\mathrm{V}, \mathrm{E} \cup \{\{\mathrm{v},\mathrm{w}\}\})$ *be the graph after adding the edge* $\{\mathrm{v},\mathrm{w}\}$*. Then*

1. *All cliques of* $\mathcal{C}(\mathrm{G})$ *that do not contain* v *or* w *are in* $\mathcal{C}(\mathrm{G}')$*.*
2. *For all* $\mathrm{A} \in \mathcal{C}(\mathrm{v},\mathrm{G})$ *and for all* $\mathrm{B} \in \mathcal{C}(\mathrm{w},\mathrm{G})$ *we have*
   (a) $L = \mathrm{A} \cap \mathrm{B} \cup \{\mathrm{v},\mathrm{w}\}$ *is a clique candidate for* G'*.*
   (b) $|\mathrm{A} \setminus \mathrm{B}| = 1 \implies \mathrm{A} \notin \mathcal{C}(\mathrm{G}')$*; otherwise* A *is a clique candidate for* G'*.*
   (c) $|\mathrm{B} \setminus \mathrm{A}| = 1 \implies \mathrm{B} \notin \mathcal{C}(\mathrm{G}')$*; otherwise* B *is a clique candidate for* G'*.*
3. *The set* $\mathcal{C}(\mathrm{G}')$ *is fully determined by statement (1) and by inspecting all the clique candidates of statement (2).*

Stix' algorithm with several improvements is presented as Algorithm 2. Notice that the first part of condition 2(b) and 2(c) is not checked in Algorithm 2. We conducted experiments with these conditions being checked, but found it to be about 40% slower. Furthermore, we accumulate clique candidates in a set to reduce the number of calls to IsClique($\cdot$).

Next we illustrate how Stix' algorithm work in a small example.

Example 1. Consider the graph in Figure 2 on the left which we want to update with the set of edges $\{\{3,4\},\{3,5\}\}$. When adding the edge $\{3,4\}$, line 7-13 in Stix' algorithm combines the two sets of cliques $\mathcal{C}(3,\mathrm{G}) = \{\{1,3\},\{3,6\}\}$

Algorithm 1 Verifying a complete set C is a clique (improved version)

```
1: function IsClique(C, G)
2:     Input: A non-empty, complete set of
3:            vertices C, and a graph G.
4:     for all v ∈ nb(C, G) do
5:         if C ⊆ nb(v, G) then
6:             return false
7:         end if
8:     end for
9:     return true
10: end function
```

and $\mathcal{C}(4, \mathrm{G}) = \{\{2,4,5\},\{4,5,6\}\}$. The resulting set of clique candidates is then $\mathcal{K} = \{\{3,4\},\{3,4\},\{3,4\},\{3,4,6\}\}$. Then follows a series of calls to `IsClique(·)` which determines that the clique $\{3,6\}$ must be removed from and the clique $\{3,4,6\}$ must be added to $\mathcal{C}$.

In the second iteration we add the edge $\{3,5\}$ and get the set of candidates $\mathcal{K} = \{\{3,5\},\{3,5\},\{3,4,5\},\{3,4,5,6\}\}$ and determine that the cliques $\{3,4,6\}$ and $\{4,5,6\}$ must be removed and the clique $\{3,4,5,6\}$ must be added.

The above example shows that there are two potential performance problems with Stix' approach when adding multiple edges:

1. Many duplicate clique candidates are generated and existing cliques are combined multiple times, and
2. A great number of calls to `IsClique(·)` is needed to prune candidates and remove old cliques.

To overcome these problems, one might try to generalize Stix' theoretical results to account for a larger set of edges being added at one time. However, it turns out that such an approach suffers even more from the problems above. In the following we shall therefore present a radically different approach that overcomes both problems.

3 Clique Maintenance by Local Search

The general idea behind this method is simple: instead of running the Bron-Kerbosch algorithm (or a similar algorithm) on the whole graph, run it on a smaller subgraph where all the new cliques appear and existing cliques disappear. Then simply update the set of cliques based on the vertices of the subgraph and the newly found cliques. Algorithm 3 is the modified Bron-Kerbosch algorithm which by using a pivot can reduce the search space (to get the original algorithm simply exchange the iteration in line 7 with "**for all** v ∈ P **do**"). In our implementation we pick the pivot deterministically as the first vertex in P because this is very easy (for alternative pivot selection strategies see (Cazals and Karande, 2008) and (Koch, 2001)).

To find all cliques of a graph G, the algorithm should be called with the argument tuple $(\mathrm{G}, \emptyset, \mathcal{V}(\mathrm{G}), \emptyset)$. However, an important ob-

Algorithm 2 Incremental clique maintenance by single-edge updates (improved version)

```
1: function EdgeBasedUpdate(C, G, F)
2:     Input: A graph G = (V, E),
3:            the set of cliques C of G, and
4:            the set of new edges F.
5:     for all {u, v} ∈ F do
6:         Let G' = (V, E(G) ∪ {{u, v}})
7:         Set K = ∅
8:         for all A ∈ C(u, G) do
9:             for all B ∈ C(v, G) do
10:                Let C = A ∩ B ∪ {u, v}
11:                Set K = K ∪ {C}
12:            end for
13:        end for
14:        for all K ∈ C(u, G) ∪ C(v, G) do
15:            if !IsClique(K, G') then
16:                Set C = C \ {K}
17:            end if
18:        end for
19:        for all K ∈ K do
20:            if IsClique(K, G') then
21:                Set C = C ∪ {K}
22:            end if
23:        end for
24:        Set G = G'
25:    end for
26:    return C
27: end function
```

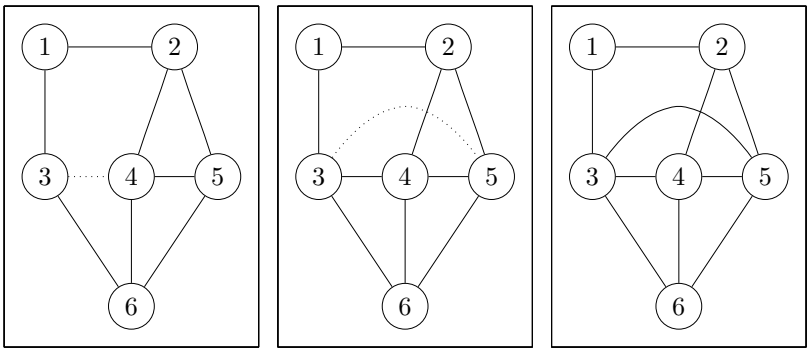


Figure 2: The sequence of graphs considered by Stix' algorithm (Algorithm 2) when adding the set of edges $\{\{3,4\},\{3.5\}\}$. Left: The initial graph—a dotted edge indicates it is about to be added to the graph. Middle: the graph after the first edge has been added. Right: The final graph.

Algorithm 3 The Bron-Kerbosch algorithm with pivot

```
1: function BKWithPivot(G, R, P, X)
2:     if P = ∅ and X = ∅ then
3:         return {R}
4:     else
5:         Let C = ∅
6:         Let u = SelectPivot(P, X, G)
7:         for all v ∈ P \ nb(u, G) do
8:             Set P = P \ {v}
9:             Let R_new = R ∪ {v}
10:            Let P_new = P ∩ nb(v, G)
11:            Let X_new = X ∩ nb(v, G)
12:            Let K =
13:        BKWithPivot(G, R_new, P_new, X_new)
14:            Set X = X ∪ {v}
15:            Set C = C ∪ K
16:        end for
17:        Return C
18:    end if
19: end function
```

servation is that the algorithm can also search a subgraph G[W] for cliques by simply passing the arguments $(G,\emptyset,W,\emptyset)$. It is this ability that our new clique maintenance algorithm takes advantage of. Our new algorithm for dynamic clique maintenance is presented in Algorithm 4, and explained in the next example.

Example 2. Consider again Figure 2. We immediately update the graph $G = (V,E)$ with the set of new edges $\{\{3,4\},\{3,5\}\}$ (line 6). The set U becomes $\{3,4,5\}$ and $\mathrm{fa}(U,G')$ actually equals V and so we will run Bron-Kerbosch on the whole graph (of course, for larger graphs this is rarely the case). Then we iterate through the existing cliques and remove those that intersect with U (line 9-13)—this step only leaves the clique $\{1,2\}$ in $\mathcal{C}$. Then we add all the cliques found in the subgraph $G'[\mathrm{fa}(U,G')]$ (line 14-18) if and only if they intersect with U—in this case only $\{1,2\}$ is not added. We can observe that the clique $\{2,4,5\}$ is both removed and added again by the algorithm.

As the above example explains, our algorithm sometimes removes and adds the same clique. This is usually not a problem in practice, as comparison of cliques is much faster than calling `IsClique(·)`. The correctness of the algorithm follows from the results below.

Lemma 1. *Let* $G = (V,E)$ *be an undirected graph, and let* $G' = (V, E \cup F)$ *be the graph resulting from adding a set of new edges* F *to* G. *Let* $U = \mathcal{V}(F)$. *If* $C \in \mathcal{NC}(G,G')$, *then* $C \subseteq \mathrm{fa}(U,G')$.

Proof. Since C is a new clique, it must contain at least two vertices from U. Since C is complete all vertices $v \in C \setminus U$ must be connected to some vertex in U. Hence v is a neighbour of U. □

Lemma 2. *Let* G *and* G' *be given as in Lemma 1. Then* $C \in \mathcal{RC}(G,G')$ *if and only if there exists* $K \in \mathcal{NC}(G,G')$ *such that* $C \subset K$.

Algorithm 4 Incremental clique maintenance by local search

```
1: function SetBasedUpdate(C, G, F)
2:     Input: A graph G = (V, E),
3:            the set of cliques C of G, and
4:            the set of new edges F.
5:     Let U = V(F)
6:     Let G' = (V, E ∪ F)
7:     Let C^new =
8:         BKWithPivot(G', ∅, fa(U, G'), ∅)
9:     for all C ∈ C do ▷ Remove old cliques
10:        if C ∩ U ≠ ∅ then
11:            Set C = C \ {C}
12:        end if
13:    end for
14:    for all C ∈ C^new do ▷ Add new cliques
15:        if C ∩ U ≠ ∅ then
16:            Set C = C ∪ {C}
17:        end if
18:    end for
19:    return C
20: end function
```

Proof. By Lemma 1, all new cliques $K \subseteq fa(U, G')$. Since a newly formed clique K is the only way to remove an existing clique C from $\mathcal{C}(G')$, the result follows. □

Theorem 2. *Let* G, G', F *and* U *be given as in Lemma 1. The cliques of* $\mathcal{C}(G')$ *can be found by removing the cliques from* $\mathcal{C}(G)$ *that intersect with* U *and adding cliques of* $G'[fa(U, G')]$ *that intersect with* U.

Proof. We first show we add all new cliques by considering just $G'[fa(U, G')]$. By Lemma 1, this subgraph contains all the new cliques. Furthermore, any new clique C must intersect with U; otherwise it could not contain a new edge. Therefore all the new cliques are added.

We remove all relevant cliques if $C \in \mathcal{RC}(G, G')$ implies $C \cap U \neq \emptyset$. So let $C \in \mathcal{RC}(G, G')$. Assume $C \cap U = \emptyset$; then for each $v \in nb(C, G) \cap U$, $C \not\subseteq nb(v, G)$ (otherwise C could not be a clique in G). But then no new clique can cover C which is a contradiction to Lemma 2. Hence $C \cap U \neq \emptyset$, and therefore we remove all relevant cliques.

Last we consider that we might also remove a clique $C \in \mathcal{C}(G) \cap \mathcal{C}(G')$. Since $C \cap U \neq \emptyset$, then $C \subseteq fa(U, G')$, and C will be added again when we add cliques from $G'[fa(U, G')]$ that intersect with U. □

Remark. Theorem 2 also implies that we can apply further pruning based on the set U in Algorithm 3. In particular, the for-loop in line 7 can be skipped if $(R \cup P) \cap U = \emptyset$. We have not implemented this pruning, however.

Remark. Stix derives a second approach to deal with the decremental problem. A nice property of our approach is that it works almost unaltered for this case (simply remove the set F of edges from the graph instead of adding them).

4 Triangulation by Clique Maintenance

An undirected graph is *triangulated* (or chordal) if every cycle of length greater than 3 has a chord. For example, the graphs in Figure 1 (left) and Figure 2 (right) are not triangulated, whereas the graph in Figure 1 (right) is triangulated. Triangulated graphs appears in remarkably diverse set of applications, ranging from efficient Gaussian elimination and compression in databases to compilation of Bayesian networks and decision graphs. It is the latter topic we have in mind in the following discussion. First we need some additional definitions.

The *elimination* of a vertex $v \in V$ of $G = (V, E)$ is the process of removing v from G and making $nb(v, G)$ a complete set. This process induces a new graph $H = (V \setminus \{v\}, E \cup F)$ where F is a set of *fill-in edges*. An *elimination order* of $G = (V, E)$ is a bijection $f : V \to \{1, 2, \ldots, |V|\}$ prescribing an order for eliminating all vertices of G. The *table size* of a clique C is given by $ts(C) = \prod_{v \in C} |sp(v)|$ where sp(v) denotes the state space of the variable corresponding to v in the Bayesian network. Finally, the *total table size* of a graph H is given by $tts(H) = \sum_{C \in \mathcal{C}(H)} ts(C)$.

Triangulation algorithms aim at minimizing different criteria. Two common criteria are the treewidth and the total table size criteria. The *treewidth* of a graph is the size of the largest

clique minus one, and the treewidth criterion requires the triangulated graph to have minimum treewidth. The total table size criteria requires the triangulated graph to have the minimum total table size. Of the two, the total table size criterion yields the most exact bound on the time and memory requirement of the subsequent inference in Bayesian networks (in particular when the domains of the variables have different size). In this section we shall therefore discuss how one may exploit knowledge of the cliques of a graph to perform a triangulation with minimum total table size. (This is an NP-hard problem (Wen, 1990), but for probabilistic inference we are in the fortunate situation that the task can often be performed off-line).

It is well-known that if all vertices are eliminated in G according to an elimination order f, the union of all the fill-in edges produced induces a triangulation of G. In this way each triangulation T of G corresponds to at least one elimination order, and we may explore the space of all possible triangulations $\mathcal{T}(\mathrm{G})$ by investigating all possible elimination orders. Even though the search space is of size $O(|\mathrm{V}|!)$, coalescing applies and reduces the search space to $O(2^{|\mathrm{V}|})$ size which turns out to be tractable for medium-sized models (say $|\mathrm{V}| \leq 64$) (Ottosen and Vomlel, 2010).

To perform this exploration, we may generate the search graph dynamically (on-demand) where each node corresponds to a subset of V being eliminated from G, and where each edge corresponds to a particular vertex being eliminated. (We use the term "node" exclusively for vertices in the search graph, and the term "vertices" exclusively for vertices in the graph being triangulated.) In the *start node* s no vertices have been eliminated, and in a *goal node* t all vertices have been eliminated and the graph G has been triangulated. To compute a cost for each path in the search graph, we associate the following with each node n:

1. $\mathrm{H} = (\mathrm{V}, \mathrm{E} \cup \mathrm{F})$: the original graph with all fill-in edges F accumulated along the path to n from the start node s.
2. The set of cliques for H, $\mathcal{C}(\mathrm{H})$.
3. The total table size for the graph H.

To maintain tts(H) efficiently we need $\mathcal{C}(\mathrm{H})$ which in turn requires H (as described in Section 3). The following example shows how one particular path in the search space is generated.

Example 3. Consider the graphs in Figure 3 where fill-in edges are indicated with dotted lines. We follow an elimination order starting with $\langle 6, 4 \rangle$. The cliques of the initial graph may be computed using the Bron-Kerbosch algorithm (Algorithm 3), and the total table size is (assuming binary variables) $3 \cdot 2^2 + 2 \cdot 2^3 = 28$ which is an lower estimate of the total table size of the triangulated graph. This information is then associated with the start node s.

Then we can make vertex 6 simplicial and run the clique update algorithm such the set of cliques is up-to-date. Again we can recompute the total table size. This information is then associated with a successor node n of s and the edge between them is labelled with vertex 6.

When we generate a successor node m of n (corresponding to the elimination of vertex 4), we must not add fill-ins to already eliminated vertices. Therefore the relevant graph corresponds to Figure 3 (right) including the fill-in edge (in particular, $\{2, 6\}$ should not be a fill-in edge). In this manner one may continue until the graph is triangulated. In this case, this happens after we eliminate vertex 4. Finally, we see that the cliques of the triangulated graph are $\{3, 4, 5, 6\}$, $\{2, 3, 4, 5\}$, and $\{1, 2, 3\}$, thus its true total table size equals $2 \cdot 2^4 + 2^3 = 40$.

5 Results

In this section we shall compare the two approaches for dynamic clique maintenance. For that purpose we have used a set of public Bayesian networks[1]. This gave us 7 real-world undirected graphs, and for each graph we generated 10 more by successively adding 5% of the missing (undirected) edges at random (in total 77 graphs). We have then performed two tests on this dataset:

[1] `http://compbio.cs.huji.ac.il/Repository/networks.html`

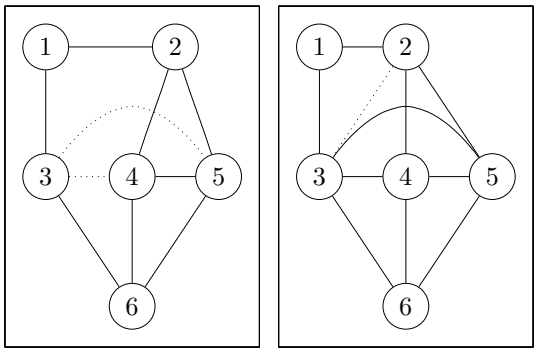


Figure 3: Left: The initial graph G = (V, E): when we eliminate vertex 6, we need to add the fill-ins $\{3, 4\}$ and $\{3, 5\}$. Right: The graph after adding fill-in edges induced by eliminating vertex 6: when we eliminate vertex 4, the fill-in $\{2, 3\}$ is added and the graph is now triangulated.

1. For each graph in the dataset, add all the missing edges in isolation. The set of cliques is updated after an edge is added. Then the edge is removed, and the next edge is added etc.
2. For each graph in the dataset, triangulate the graph by making all vertices simplicial in some random order. The set of cliques is updated after each vertex is made simplicial. Already simplicial vertices are removed before an update. (Note that we did not moralize the initial directed network even though this might lead to a slightly more accurate test.)

These two test scenarios were chosen because we believe that they show the worst-case performance (scenario 1), and the expected speedup for our triangulation problem (scenario 2). For each graph we then ran the tests 1000 times (with different random order each time for Test 2) and saved the mean time. We have then plotted the mean time of Stix' approach divided by the mean time of the new approach (we call this the "saving ratio").

In Figure 4 and 5 we have collected the results of the two tests. The total time saving ratio for various graph densities are summarized in Table 1. We can see that even for Test 1, the new

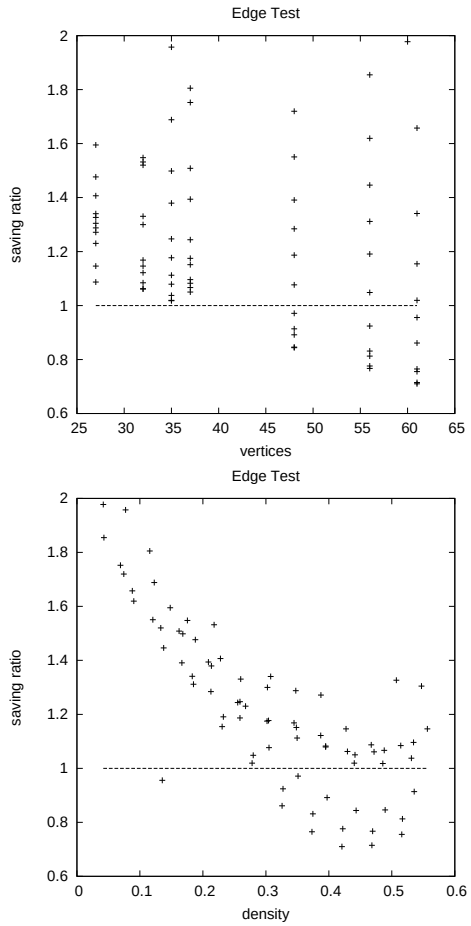


Figure 4: Results for Test 1: updating the set of cliques after adding a single edge. Points above $y = 1$ indicate that the new approach is faster.

method often performs better. However, overall Stix' method works better for more dense graphs. There seems to be no clear connection between the size of the graph and saving ratio.

For Test 2 the new method is significantly better, especially for more dense graphs. There also seems to be a connection between the size of the graph and performance, with the saving ratio increasing as the size increases. This might be because larger graphs allow for more cliques and larger neighbour sets.

6 Conclusion

We have described a new method for maintaining the cliques of a dynamic graph. The new method works by employing a local search for cliques—the local search can in principle

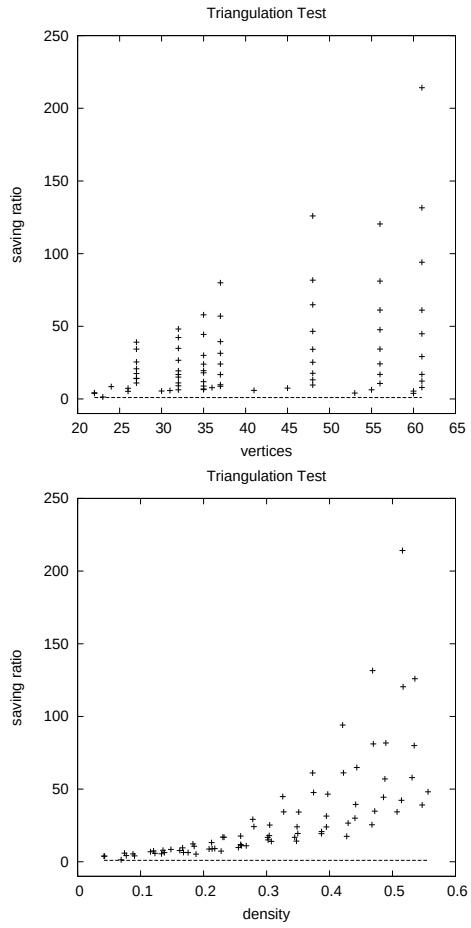


Figure 5: Results for Test 2: updating the set of cliques after adding fill-in edges. Points above $y = 1$ indicate that the new approach is faster.

be done by any existing clique search algorithm. The new method is both simpler and more generic than previous methods, and experiments show that the new method performs significantly faster when adding a set of fill-in edges.

We also described how dynamic clique maintenance algorithms may found the basis for new total table size triangulations methods. These methods may be optimal off-line triangulations, or they may be any-time triangulation heuristics. Since the off-line triangulations methods can explore the whole space of possible triangulations, they can also be used to give a precise evaluation of the quality of the triangulations returned by heuristics.

Table 1: Total time saving ratio for different graph densities. A value above 1 indicates that the new method was faster in terms of total running time for all graphs of the specified density.

Density δ	Test 1	Test 2
$\delta \in [0, 0.1)$	1.74	4.77
$\delta \in [0.1, 0.2)$	1.32	9.09
$\delta \in [0.2, 0.3)$	1.12	19.35
$\delta \in [0.3, 0.4)$	0.88	40.95
$\delta \in [0.4, 0.5)$	0.76	86.68
$\delta \in [0.5, 0.6)$	0.80	153.04

Acknowledgements

We would like to thank the three anonymous reviewers for their constructive comments.

J. Vomlel was supported by the Ministry of Education of the Czech Republic through grants 1M0572 and 2C06019 and by the Czech Science Foundation through grants ICC/08/E010 and 201/09/1891.

References

F. Cazals and C. Karande. 2008. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1-3):564–568, November.

R. M. Karp. 1972. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Ina Koch. 2001. Enumerating all connected maximal common subgraphs in two graphs. *Theor. Comput. Sci.*, 250(1-2):1–30.

J. W. Moon and L. Moser. 1965. On cliques in graphs. *Israel Journal of Mathematics*, 3:23–28.

Thorsten J. Ottosen and Jiří Vomlel. 2010. All roads lead to Rome—new search methods for optimal triangulations. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*.

Volker Stix. 2004. Finding all maximal cliques in dynamic graphs. *Comput. Optim. Appl.*, 27(2):173–186.

Wilson Wen. 1990. Optimal decomposition of belief networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 209–224, New York, NY. Elsevier Science.

All Roads Lead To Rome—New Search Methods for Optimal Triangulation

Thorsten J. Ottosen
Department of Computer Science, Aalborg University, Denmark
nesotto@cs.aau.dk

Jiří Vomlel
Institute of Information Theory and Automation of the AS CR, The Czech Republic
vomlel@utia.cas.cz

Abstract

To perform efficient inference in Bayesian networks, the network graph needs to be triangulated. The quality of this triangulation largely determines the efficiency of the subsequent inference, but the triangulation problem is unfortunately NP-hard. It is common for existing methods to use the treewidth criterion for optimality of a triangulation. However, this criterion may lead to a somewhat harder inference problem than the total table size criterion. We therefore investigate new methods for depth-first search and best-first search for finding optimal total table size triangulations. The search methods are made faster by efficient dynamic maintenance of the cliques of a graph. The algorithms are mainly supposed to be off-line methods, but they may form the basis for efficient any-time heuristics. Furthermore, the methods make it possible to evaluate the quality of heuristics precisely.

1 Introduction

We consider the problem of finding optimal cost triangulations of Bayesian networks. We solve this problem by searching the space of all possible triangulations. This search is carried out by trying all possible elimination orders and choosing one of those that have a minimal total table size. Of all commonly-used optimality criteria, the total table size yields the most exact bound of the memory and time requirement of the probabilistic inference. However, finding optimal triangulations is difficult: computing a minimum fill-in is NP-complete (Yannakakis, 1981) and finding a triangulation with minimal total table size is NP-hard (Wen, 1990).

There are several issues that motivates an investigation of this problem. Since the problem is NP-hard, we cannot expect the problem to be solvable in a reasonable amount of time for large networks. However, triangulation can always be performed off-line on specialized servers and saved for use by the inference algorithms. This is important as intractability or simply poor performance is a major obstacle to more wide-spread adoption of Bayesian networks and decision graphs in statistics, engineering and other sciences. Furthermore, efficient off-line algorithms allow us to evaluate the quality of on-line methods which can otherwise only be compared to other on-line methods. An off-line method, on the other hand, can effectively answer whether the subsequent inference is tractable. Finally, off-line methods can often be turned into good any-time heuristics.

Previous research on triangulation has also used best-first search (Dow and Korf, 2007) and depth-first search (Gogate and Dechter, 2004), however, the optimality criteria is the treewidth of the graph and so the found triangulation is (in the best case) only guaranteed to be within a factor of n (n being the number of vertices of the graph) from the optimal total table size triangulation—this factor could mean the difference between an intractable and a tractable inference. With the treewidth optimality cri-

Pp. 209–217 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

terion, one can continuously apply the preprocessing rules of (Bodlaender et al., 2005), but for the total table size criterion we can (so far) only remove simplicial vertices which makes this problem considerably harder. The seminal idea of divide-and-conquer triangulating using decomposable subgraphs dates back to (Tarjan, 1985). Leimer refines this approach such that the generated subgraphs are not themselves decomposable (i.e., they are maximal prime subgraphs and this unique decomposition is denoted a maximal prime subgraph decomposition) (Leimer, 1993). Basically, this means that the problem of triangulating a graph G is no more difficult than triangulating the largest maximal prime subgraph of G. This decomposition is exploited in (Flores and Gámez, 2003).

In (Shoikhet and Geiger, 1997) a dynamic programming algorithm is given based on decompositions by minimal separators, and again the optimality criterion is treewidth. As noted by its authors, the method may be adopted to yield an optimal total table size triangulation as well. Finally, an overview of triangulation approaches is given in (Flores and Gámez, 2007).

2 Preliminaries

We shall use the following notation and definitions. $G = (V, E)$ is an *undirected graph* with *vertices* $V = \mathcal{V}(G)$ and *edges* $E = \mathcal{E}(G)$. For a set of edges F, $\mathcal{V}(F)$ is the set of vertices $\{u, v : \{u, v\} \in F\}$. An undirected graph is *triangulated* (or chordal) if every cycle of length greater than 3 has a chord. For example, in Figure 1 the graph on the left is not triangulated whereas the graph on the right is triangulated. For $W \subseteq V$, $G[W]$ is the *subgraph induced by* W. A *triangulation* of G is a set of edges T such that $T \cap E = \emptyset$ and the graph $H = (V, E \cup T)$ is triangulated. We denote the set of all triangulations of a graph G for $\mathcal{T}(G)$.

Two vertices u and v are *connected* in G if there is an edge between them. A graph G is *complete* if all pairs of vertices $\{u, v\}$ $(u \neq v)$ are connected in G. A set of vertices $W \subseteq V$ is *complete in* G if $G[W]$ is a complete graph. The *neighbours* $nb(v, G)$ of a vertex $v \in V$ is the set $W \subseteq V$ such that each $u \in W$ is connected to v. The *family* $fa(v, G)$ of a vertex v is the set $nb(v, G) \cup \{v\}$, and the neighbours and family of a set of vertices is defined similarly. The *elimination* of a vertex $v \in V$ of $G = (V, E)$ is the process of removing v from G and making $nb(v, G)$ a complete set. This process induces a new graph $H = (V \setminus \{v\}, E \cup F)$ where F is the set of *fill-in edges*. For example, in Figure 1 (left), eliminating the vertex 6 induces the two fill-in edges shown with dotted edges in the adjacent graph. If $F = \emptyset$, then v is a *simplicial vertex*. An *elimination order* of $G = (V, E)$ is a bijection $f : V \to \{1, 2, \ldots, |V|\}$ prescribing an order for eliminating all vertices of G. If all vertices are eliminated in G according to an elimination order f, the union of all the fill-in edges produced induces a triangulation of G. In this way each triangulation T of G corresponds to at least one elimination order, and we may explore the space $\mathcal{T}(G)$ by investigating all possible elimination orders.

Given $G = (V, E)$, a set of vertices $C \subseteq V$ is a *clique* if it is a maximal complete set and $\mathcal{C}(G)$ is the set of all cliques in G. The *table size* of a clique C is given by $ts(C) = \prod_{v \in C} |sp(v)|$ where $sp(v)$ denotes the state space of the variable corresponding to v in the Bayesian network. Finally, the *total table size* of a graph H is given by $tts(H) = \sum_{C \in \mathcal{C}(H)} ts(C)$.

Triangulation algorithms aim at minimizing different criteria. The most common are the fill-in, the treewidth and the total table size criteria. The *fill-in criterion* requires the triangulated graph to have the minimum total number of fill-in edges. The *treewidth* of a graph is the size of the largest clique minus one, and the *treewidth criterion* requires the triangulated graph to have minimum treewidth. The *total table size criteria* requires the triangulated graph to have the minimum total table size. Commonly seen triangulation heuristics include min-fill and min-width which both greedily pick the next vertex to eliminate based on a local score. In *min-fill* a vertex is chosen if its elimination leads to the fewest fill-in edges; in *min-width* a vertex is chosen if it has the fewest number of neighbours.

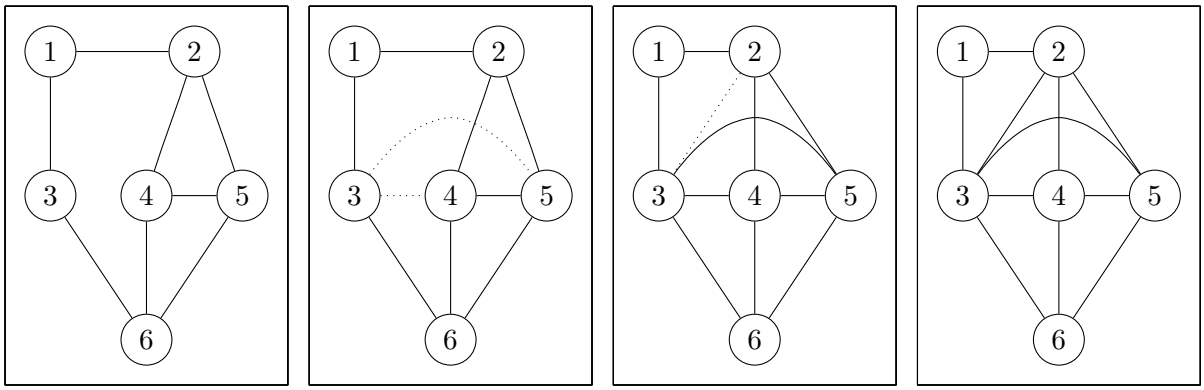


Figure 1: Example of the fill-in edges and partially triangulated graphs induced by an elimination order that starts with the sequence $\langle 6, 4 \rangle$: the dotted edges are fill-in edges. Left: the initial graph. Middle left: the fill-in edges induced by eliminating vertex 6. Middle right: the fill-in edges induced by eliminating vertex 4. Right: the final triangulated graph.

3 The Search Space for Triangulation Algorithms

Our goal is to explore the space $\mathcal{T}(\mathrm{G})$ encoding all possible ways to triangulate a graph G in. To do this, we generate a search graph dynamically (on-demand) where each node corresponds to a subset of V being eliminated from G, and where each edge is labelled with the particular vertex that has been eliminated. (Note that we exclusively use the term "node" for vertices in the search graph whereas the term "vertex" is used exclusively for vertices in the undirected graph being triangulated.) In the *start node* s no vertices have been eliminated, and in a *goal node* t all vertices have been eliminated and the graph G has been triangulated.

Since we are seeking optimal total table size triangulations we also need to associate this quantity with each node. By definition, the total table size is easy to compute if we know the cliques of the partially triangulated graph, and therefore we also need to associate this graph with each node. Below we give a small example of a path in the search space—in Section 5 we shall explain the algorithms in detail.

Example 1. Consider the graphs in Figure 1 and assume that all variables (in the original Bayesian network) are binary. The graph associated with the start node s would be the graph on the left and this graph has a total table size of $2^2 + 2^2 + 2^2 + 2^3 + 2^3 = 28$.

The graph associated with a successor node m of s (corresponding to the elimination of vertex 6) would correspond to the graph in the middle (left) (including dotted edges) with total table size $2^2 + 2^2 + 2^3 + 2^4 = 32$.

And the successor node of m (corresponding to the elimination of vertex 4) would be associated with graph on the right, which is also a goal node, with total table size $2^3+2^4+2^4 = 40$. Note that when introducing fill-in edges, we must not add edges to vertices that has already been eliminated—this is why this step does not add the edge $\{2, 6\}$ even though the vertices are both neighbours of vertex 4.

Observe that the total table size of a node is never higher than the total table size of its successor node(s). This implies that the total table size associated with any non-goal node n is a lower-bound on the total table size of any goal node that may be discovered from n. This property guarantees that the algorithms in Section 5 are admissible.

4 Dynamic Clique Maintenance

To compute the cliques of a graph associated with a node in the search graph, we may use a standard algorithm for this task, for example, the well-known Bron-Kerbosch algorithm (Cazals and Karande, 2008). However, as we

can see from the above example, each path in the search graph corresponds to a sequence of graphs where the difference between adjacent graphs is quite small. Therefore we may exploit this similarity among adjacent graphs to avoid the quite expensive recomputation of the cliques and total table size of the graphs.

(Stix, 2004) investigated this problem, however, his method leads to many redundant computations when the added edges appear close together (as is the case for fill-in edges). Therefore we give a new algorithm that performs significantly faster for this type of update.

The general idea behind this method is simple: instead of searching for cliques in the whole graph, simply run a clique enumeration algorithm on a smaller subgraph where all the new cliques appear and existing cliques disappear. This algorithm for dynamic clique maintenance is presented as Algorithm 1, and as a side-effect it also updates the total table size and the current graph. This implies that the total table size does not need to be computed from scratch either. In our case we employ Bron-Kerbosch for the local search in `FindCliques(·)`. Its correctness follows from the following result.

Theorem 1. (Ottosen and Vomlel, 2010). *Let* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ *be an undirected graph, and let* $\mathrm{G}' = (\mathrm{V}, \mathrm{E} \cup \mathrm{F})$ *be the graph resulting from adding a set of new edges* F *to* G. *Let* $\mathrm{U} = \mathcal{V}(\mathrm{F})$. *The cliques of* $\mathcal{C}(\mathrm{G}')$ *can be found by removing the cliques from* $\mathcal{C}(\mathrm{G})$ *that intersect with* U *and adding cliques of* $\mathrm{G}'[\mathrm{fa}(\mathrm{U}, \mathrm{G}')]$ *that intersect with* U.

(Xiang and Lee, 2006) describes a set of vertices called a *cruz* which is central to their method for learning. The method described above may also be used to efficiently determine the cruz.

5 Optimal Total Table Size Triangulation Algorithms

We have now shown how we may efficiently compute the total table size for each successor m of a node n in the search space $\mathcal{T}(\mathrm{G})$, and we have furthermore established that the total table size for a node n is a lower-bound of any possible triangulation associated with the set of goal

Algorithm 1 Incremental maintenance of cliques and total table size by local search

```
1:  procedure IncrUpdate(&G, &C, &tts, F)
2:      Input: A graph G = (V, E),
3:             the set of cliques C of G,
4:             the total table size tts of G, and
5:             the set of new edges F.
6:      Set G = (V, E ∪ F)
7:      Let U = V(F)
8:      Let C^new = FindCliques(G, fa(U, G))
9:      for all C ∈ C do  ▷ Remove old cliques
10:         if C ∩ U ≠ ∅ then
11:             Set tts = tts - ts(C)
12:             Set C = C \ {C}
13:         end if
14:     end for
15:     for all C ∈ C^new do  ▷ Add new cliques
16:         if C ∩ U ≠ ∅ then
17:             Set tts = tts + ts(C)
18:             Set C = C ∪ {C}
19:         end if
20:     end for
21: end procedure
```

node reachable from n. Given this, we may use standard algorithms like best-first search and depth-first search to explore the search space and at the same time be guaranteed that the algorithms terminate with an optimal solution.

Best-first search is an algorithm that successively expands nodes with the shortest distance to the start node until a goal node has a shorter path than all non-goal nodes. The benefit of the best-first strategy is that we may avoid exploring paths that are far from the optimal path. The disadvantage of a best-first strategy is that the algorithm must keep track of a *frontier* (or fringe) or nodes that still needs to be explored. *Depth-first search*, on the other hand, explores all paths in a depth-first manner and therefore uses only $\Theta(|\mathrm{V}|)$ memory for a graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$. However, depth-first search is typically forced to explore more paths than best-first search.

To compute a cost for each path in the search graph, we associate the following with each node n:

1. $\mathrm{H} = (\mathrm{V}, \mathrm{E} \cup \mathrm{F})$: the original graph with all fill-in edges F accumulated along the path to n from the start node s.
2. R: the remaining graph $\mathrm{H}[\mathrm{V} \setminus \mathrm{W}]$ where W are the vertices of G eliminated along the path from s to n.
3. $\mathcal{C}$: the set of cliques for H, $\mathcal{C}(\mathrm{H})$.
4. tts: the total table size for the graph H.
5. $\mathcal{L}$: a list of vertices describing the elimination order.

To maintain tts(H) efficiently we need $\mathcal{C}(\mathrm{H})$ which in turn requires H, and we saw how this can be done in Section 4 . The graph R makes it easy to determine if the graph H is triangulated and may be computed on demand to reduce memory requirements.

In the worst case, the complexity of any best-first search method is $O(\beta(|\mathrm{V}|) \cdot |\mathrm{V}|!)$ (where $\beta(\cdot)$ is a function that describes the per-node overhead) because we must try each possible elimination order. However, it is well known that the remaining graph $\mathrm{H}[\mathrm{V} \setminus \mathrm{W}]$ is the same no matter what order the vertices in W have been eliminated in, so we can use *coalescing* of nodes and thus reduce the worst case complexity to $O(\beta(|\mathrm{V}|) \cdot 2^{|\mathrm{V}|})$ (Darwiche, 2009).

For depth-first search the complexity is often thought to remain at $\Theta(\gamma(|\mathrm{V}|) \cdot |\mathrm{V}|!)$, however, at the expense of memory we may also apply coalescing for pruning purposes. Hence, depth-first search can be made to run in $O(\gamma(|\mathrm{V}|) \cdot |\mathrm{V}|!)$ time using $O(2^{|\mathrm{V}|})$ memory, but the hidden constants will be much smaller in this case compared to best-first search.

Both $\gamma(|\mathrm{V}|)$ and $\beta(|\mathrm{V}|)$ take at least $O(|\mathrm{V}|^3)$ time as they are dominated by the removal of simplicial vertices (the lookup into the coalescing map takes $O(|\mathrm{V}|)$ time due to the computation of the hash-key, and the priority queue look-up for best-first search may take $O(|\mathrm{V}|)$ time since the queue may become exponentially large). Getting a more precise bound on the two functions is difficult as the complexity of maintaining the cliques and total table size depends very much on the graph being triangulated.

In Algorithm 2 we describe the basic best-first search with coalescing, and depth-first search with coalescing and pruning based on the currently best path is described in Algorithm 3. The procedure `EliminateVertex(·)` simply eliminates a vertex from the remaining graph R and updates the cliques and total table size of the partially triangulated graph H (see Section 4). The procedure `EliminateSimplicial(·)` removes all simplicial vertices from the remaining graph.

Algorithm 2 Best-first search with coalescing

```
function TriangulationByBFS(G)
    Let s = (G, G, C(G), tts(G), ⟨⟩)
    EliminateSimplicial(s)
    if |V(s.R)| = 0 then
        return s
    end if
    Let map = ∅                          ▷ Coalescing map
    Let O = {s}                          ▷ The open set
    while O ≠ ∅ do
        Let n = arg min_{x∈O} x.tts
        if |V(n.R)| = 0 then
            return n
        end if
        Set O = O \ {n}
        for all v ∈ V(n.R) do
            Let m = Copy(n)
            EliminateVertex(m, v)
            EliminateSimplicial(m)
            if map[m.R].tts ≤ m.tts) then
                continue
            end if
            Set O = O \ {map[m.R]}
            Set map[m.R] = m
            Set O = O ∪ {m}
        end for
    end while
end function
```

6 Results

In this section we describe experiments with the optimal methods as well as several heuristics derived from these. For that purpose we have generated 50 random graphs with varying size and density. In this paper we have only performed experiments on bipartite graphs—these graphs

Algorithm 3 Depth-first search with coalescing and upper-bound pruning

```
function TriangulationByDFS(G)
    Let s = (G, G, C(G), tts(G), ⟨⟩)
    EliminateSimplicial(s)
    if |V(s.R)| = 0 then
        return s
    end if
    Let best = MinFill(s)          ▷ Best path
    Let map = ∅                    ▷ Coalescing map
    ExpandNode(s, best, map)
    return best
end function
procedure ExpandNode(n, &best,&map)
    for all v ∈ V(n.R) do
        Let m = Copy(n)
        EliminateVertex(m, v)
        EliminateSimplicial(m)
        if |V(m.R)| = 0 then
            if m.tts < best.tts then
                Set best = m
            end if
        else
            if m.tts ≥ best.tts then
                continue
            end if
            if map[m.R].tts ≤ m.tts) then
                continue
            end if
            Set map[m.R] = m
            ExpandNode(m, best, map)
        end if
    end for
end procedure
```

result from the application of rank-one decomposition to BN2O networks—see (Savicky and Vomlel, 2009) for details. The main reason for using these graphs is that they are among the most difficult to triangulate. This is because (1) moralization should not be applied after using rank-one decomposition, and (2) bottom and top layers are not connected. Thereby the initial graph is sparser than usual which gives triangulation algorithms more freedom (in terms of choosing fill-in edges) when searching for a triangulation.

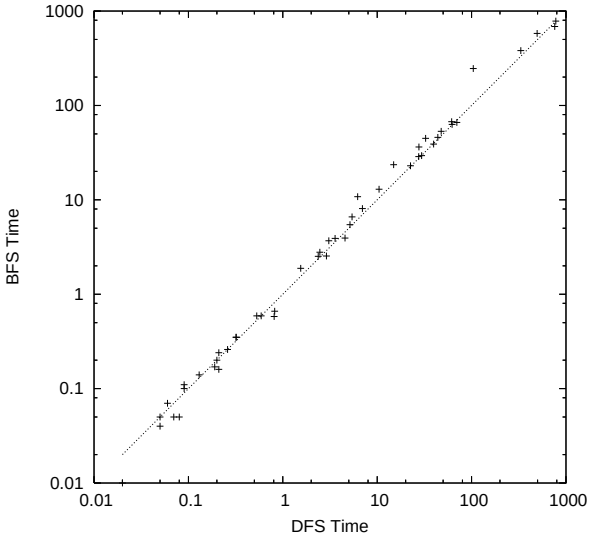


Figure 2: Comparison of the running time of best-first search and depth-first search. Both algorithms terminate with an optimal triangulation (values above the line indicate depth-first search was faster).

We have performed two different tests on this dataset: (1) a comparison of depth-first search and best-first search, and (2) a comparison between heuristic methods. For Test 2 we have implemented the following heuristic methods:

(a) *Limited-branching depth-first search.* This means that we only expand the n successors of a node which have the lowest total table size. For example, "limited-branch-5" expands at most five successors per node.

(b) *Limited memory best-first search.* Here we limit the size of the open set $\mathcal{O}$ to some fixed value n by removing the worst nodes when the set is considered full. For example, "limited-mem-10k" has at most 10.000 nodes in its open set.

(c) The min-width and min-fill heuristics implemented so that a successor node is generated for all ties instead of breaking ties randomly. We refer to these algorithms as *min-width** and *min-fill**, respectively.

The results from Test 1 are given in Figure 2. It appears that best-first search performs better when the computational time is large.

Table 1: Summary statistics for exhaustive search algorithms. Each row summarizes the mean time for graphs with n vertices. The value in parenthesis in the first column indicates the number of graphs of that size.

Vertices	DFS	BFS
20 (10)	0.3s	0.3s
30 (20)	9.1s	17.5s
40 (10)	30.6s	31.8s
50 (5)	30.4s	33.0s
60 (5)	591.7s	607.0s
% Fastest	71	37

Table 2: Results for heuristic algorithms. The first column describes the percentage of graphs that were triangulated optimally, and the second column contains the maximum percent-wise deviation from the total table size of an optimal triangulation. The third column indicates total time for triangulating all 50 graphs.

	% op.	% dev.	time
min-width*	82	23,916	1s
min-fill*	84	1,322	1s
lim.-br.-2 DFS	94	53	83s
lim.-br.-3 DFS	98	27	270s
lim.-br.-4 DFS	100	0	512s
lim.-mem-100 BFS	96	2	2234s
lim.-mem-1k BFS	100	0	6447s

In Table 1 we give summary statistics for this test. We have computed the p-value of the two-sided Wilcoxon two-sample test of the null hypothesis that the distribution of depth-first time minus best-first time is symmetric about 0. The p-value is 0.001069, which means that the null hypothesis is rejected, that is, differences are significant (in favor of depth-first search).

The results from Test 2 are given in the Table 2. Here we have run the heuristics on the 50 graphs from Test 1. From this we can conclude that min-fill and min-width are quite often good heuristics, but that their induced search spaces are too small to avoid triangulations that are far from optimal. The new heuristics seem to avoid this pitfall.

Notice that the BN2O networks only have binary variables. Therefore min-width actually corresponds to the commonly used min-weight heuristic. The graph where min-width* found an exceedingly poor triangulation has 40 vertices and a density around 0.36. The total table size for min-width* was $595,634,176$, and this means that no stochastic (breaking ties randomly) min-width heuristic can yield a triangulation that requires below some 2.4 GB of memory (assuming 4 bytes for a `float`). Contrast this with the optimal triangulation which leads to a memory requirement of only 10 MB.

7 Discussion

The fact that depth-first search came out as the fastest algorithm must be considered a surprise. We believe that the main reason for this is that the pruning via the coalescing map turns out to work quite well—this pruning is the direct cause of the change in complexity from $\Theta(\gamma(|\mathrm{V}|)\cdot|\mathrm{V}|!)$ to $O(\gamma(|\mathrm{V}|)\cdot|\mathrm{V}|!)$. The experiments indicate that best-first search actually runs in $O(\gamma(|\mathrm{V}|)\cdot 2^{|\mathrm{V}|})$ time. Secondly, it is worth mentioning that depth-first search only needs very few (otherwise expensive) free-store allocations. To further improve the pruning by the coalescing map, then we should consider a hybrid best-first-depth-first scheme where we explore the most promising paths earlier. In light of this discussion we believe depth-first search should be reconsidered also for the minimum treewidth criterion.

8 Conclusion

The contributions of this paper are three-fold. First, we have described new methods for finding optimal total table size triangulations of undirected graphs. The methods rely heavily on efficient dynamic maintenance of the cliques and total table size of a graph. These methods are mainly supposed to be used off-line, but they may also be transformed into any-time heuristics.

Secondly, experiments show that depth-first search is faster than best-first search—this was quite unexpected. The main reason is that we

use pruning based on a coalescing map which lowers the time complexity from $\Theta(\gamma(n) \cdot n!)$ to $O(\gamma(n) \cdot n!)$ (n being the number of vertices in the graph and $\gamma(n)$ being the per-node overhead). From the experiments we can infer that this pruning is so effective that depth-first search actually runs in $O(\gamma(n) \cdot 2^n)$ time. Therefore we believe it will be beneficial to reconsider depth-first search for triangulation with the minimum treewidth criterion.

Third, we have examined the quality of common heuristic algorithms on a set of graphs that are quite difficult to triangulate. The experiments show that these heuristics will never be able to guarantee good triangulations on all types of graphs, for example, on one model the min-width (and min-weight) heuristic would return a triangulation that requires at least 2.4 GB of memory whereas the optimal solution requires only 10 MB. This shows that off-line triangulation methods could be required in some cases.

Acknowledgement

The authors would like to thank the three anonymous reviewers for their helpful comments.

J. Vomlel was supported by the Ministry of Education of the Czech Republic through grants 1M0572 and 2C06019 and by the Czech Science Foundation through grants ICC/08/E010 and 201/09/1891.

References

Hans L. Bodlaender, Arie M.C.A. Koster, and Frank van den Eijkhof. 2005. Preprocessing rules for triangulation of probabilistic networks. *Computational Intelligence*, 21:286–305.

F. Cazals and C. Karande. 2008. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1-3):564–568, November.

Adnan Darwiche. 2009. *Modelling and Reasoning with Bayesian Networks*. Cambridge University Press.

P. Alex Dow and Richard E. Korf. 2007. Best-first search for treewidth. In *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*, pages 1146–1151. AAAI Press.

M. Julia Flores and José A. Gámez. 2003. Triangulation of Bayesian networks by retriangulation. *International Journal of Intelligent Systems*, 18:153–164.

M. Julia Flores and José A. Gámez. 2007. A review on distinct methods and approaches to perform triangulation for Bayesian networks. *Advances in Probabilistic Graphical Models*, pages 127–152.

Vibhav Gogate and Rina Dechter. 2004. A complete anytime algorithm for treewidth. In *Proceedings of the Proceedings of the Twentieth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 201–208, Arlington, Virginia. AUAI Press.

Hanns-Georg Leimer. 1993. Optimal decomposition by clique separators. *Discrete Math.*, 113(1-3):99–123.

Thorsten J. Ottosen and Jiří Vomlel. 2010. Honour thy neighbour—clique maintenance in dynamic graphs. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*.

Petr Savicky and Jiří Vomlel. 2009. Triangulation heuristics for BN2O networks. In C. Sossai and G. Chemello, editors, *Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 566–577. Springer.

Kirill Shoikhet and Dan Geiger. 1997. A practical algorithm for finding optimal triangulations. In *AAAI'97: Proceedings of the 14th national conference on Artificial intelligence*, pages 185–190. AAAI Press.

Volker Stix. 2004. Finding all maximal cliques in dynamic graphs. *Comput. Optim. Appl.*, 27(2):173–186.

Robert E. Tarjan. 1985. Decomposition by clique separators. *Discrete Mathematics*, 55(2):221 – 232.

Wilson Wen. 1990. Optimal decomposition of belief networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 209–224, New York, NY. Elsevier Science.

Y. Xiang and J. Lee. 2006. Learning decomposable markov networks in pseudo-independent domains with local evaluation. *Mach. Learn.*, 65(1):199–227.

Mihalis Yannakakis. 1981. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):77–79.

An Aggregation and Disaggregation Procedure for the Maintenance of a Dynamic System under Partial Observations

Demet Özgür-Ünlüakın
Bahçeşehir University, Istanbul-Turkey
demet.unluakin@bahcesehir.edu.tr

Taner Bilgiç
Boğaziçi University, Istanbul-Turkey
taner@boun.edu.tr

Abstract

We study the maintenance of a dynamic system consisting of several components each of which age with a constant transition probability of failure. The state of the components are hidden. However, partial observations exist in each time period. The decision of whether to replace a component or to do nothing is to be made in each decision epoch. A hierarchical solution procedure is proposed to solve the problem. An aggregate model is developed by aggregating states and actions so that it can be solved with exact partially observable Markov decision process (POMDP) techniques. Then disaggregation is performed by simulating the process using a dynamic Bayesian network (DBN) and applying troubleshooting approaches in the decision epoch where replacement is planned in the aggregate policy.

1 Introduction

We consider the maintenance planning problem of a dynamic system whose status is not observable but is estimated through indirect signals. Such problems are common in space systems and hazardous material detection systems where the system is far away and cannot be directly observed. However, such systems can still be controlled remotely to perform diagnostic tests and repair actions.

The major difference of the problem we consider from other maintenance problems in the literature is the complex structure of the system due to several (possibly interacting) components which results in a huge state space.

Our approach to maintenance is akin to the decision-theoretic troubleshooting problems (DTTP) (Kalagnam and Henrion, 1988) in handling complex system structures. Complex system structure in DTTP is usually represented with Bayesian Networks (BNs) (Heckerman et al., 1995; Jensen et al., 2001) which encode the conditional probabilistic dependence relations of the components. We use the same kind of BN representation at each discrete decision epoch. In DTTP the task is to find a minimum-cost action plan. DTTP has always been studied as a static problem under the assumption that a system has an observable malfunction and then troubleshooting process starts. However, we study a dynamic problem, hence the objective is to minimize the total expected costs, comprised of replacement and observation costs for a finite planning horizon.

BNs have been used in reliability problems to represent the (complex) relations in the system (Torres-Toledano and Sucar, 1998; Bobbio et al., 2001; Langseth and Portinale, 2006). Recently there have also been studies using DBNs in reliability analysis (Welch and Thelen, 2000; Weber and Jouffe, 2003; Muller et al., 2004). A DBN is an extended BN which includes a temporal dimension. However all reliability studies with DBNs are *descriptive* (i.e.,the dynamic problem is represented with DBNs and the outcome of the analysis is how system reliability behaves in time). The impact of performing a repair at a specific time on this behavior is also reported in some of them. However optimization of maintenance activities (i.e., finding a minimum cost plan) is not considered which is one of the main motivations of this paper.

Pp. 217–225 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

Therefore our approach is *prescriptive* as opposed to being descriptive.

The maintenance problem we consider can be modeled as a POMDP (Hauskrecht, 2000) in which the decision maker makes sequential decisions under partial information. POMDPs fall prey to the "curse of dimensionality" as their state space grows exponentially with the number of components in the system. The type of a POMDP model having several hidden variables is rarely studied due to its computational complexity. There are algorithms for computing optimal solutions to POMDPs. However these algorithms are applicable in practice only to relatively simple problems. Also some structural results have been presented in some specific machine maintenance applications (Ross, 1971; Rosenfield, 1976), but such results are not common for general POMDPs.

The rest of the paper is organized as follows: In Section 2, we define and represent the maintenance planning problem. In Section 3, we present the proposed solution. In Section 4, we give the experimental design and the results of computational study. In Section 5, we conclude and point to further research directions.

2 Problem Definition and Representation

2.1 Problem Definition

There is a system consisting of several components each of which age with a constant transition probability of failure. It is not possible to observe the system components, they are hidden. However the system gives signals at each decision epoch which may indicate some partial information about the state of the components. This is the only information one can get from the process. System components age with a constant rate constituting the dynamic behavior of the problem. It is possible to replace components in any period and when a component is replaced, a replacement cost is charged. On the other side, since observing a signal indicating faulty components in the next decision epoch is undesirable, it incurs another type of cost, which is called observation cost in this study. There is a trade off between replacing the components and observing undesirable signals. In each decision epoch, the decision maker can either prefer doing nothing or replacing only one of the components. The aim is to find a policy that minimizes expected total replacement and observation cost in a given horizon. The following assumptions are made:

(i) Every component has a constant transition probability of failure. (ii) All other conditional probability distributions are discrete. (iii) All components have two states ("w": working state, "nw": failure state). (iv) Components can only fail at the beginning of a time period. Once they fail they will be in state "nw" unless they are repaired. (v) Once a component is replaced in a period, its working state probability becomes 1 in the next period. (vi) Only one replacement can be done in a given epoch.

The first three assumptions are required for computational purposes as DBN tools usually work with discrete probabilities and states. The fourth and fifth assumptions are standard in reliability. The sixth assumption implies that there is a limit to replacements at each epoch (possibly due to a time or a budget constraint).

2.2 Problem Representation

The maintenance problem can be expressed as a POMDP with the following parameters:

I : number of components in the process
i : index for component
C_i: set of states of component i, $i = 1, \cdots, I$
A : set of actions
Θ : set of observations
T_i': set of transition probabilities of component i.
  T_i' :$C_i \times A \times C_i \rightarrow [0, 1]$
O': set of observation probabilities among component states and observations.
  O': $C_1 \times \cdots \times C_I \times \Theta \rightarrow [0, 1]$
R : reward function that assigns rewards to observations and actions. $R : A \times \Theta \rightarrow \mathbb{R}$

Part (only two epochs) of the influence diagram describing the problem is given in Figure 1 where c_{it}, a_t, o_t and r_t denote component state, action, observation and reward at time t.

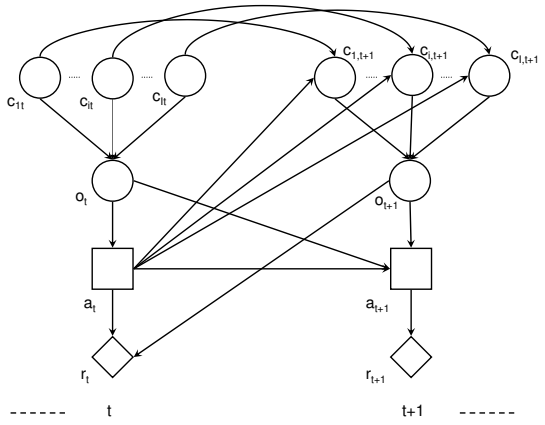


Figure 1: Part of the influence diagram describing the maintenance problem

The system gives two types of signals: g and r which stand for green and red respectively, $\Theta = \{g, r\}$. A green signal is preferred to a red signal since it implies a better condition of components. In each epoch, the decision maker can either prefer doing nothing or replacing one of the components. Hence $A = \{dn, rc_1, rc_2, \cdots, rc_I\}$, where dn denotes doing nothing and rc_i denotes replacing component i for $i = 1, 2, \cdots, I$. Table 1 shows the transition probabilities, T_i', for component i having two states, $C_i = \{w, nw\}$, given $a_t \in A \backslash \{rc_i\}$ and $a_t = rc_i$ respectively. Here, $a_t \in A \backslash \{rc_i\}$ means the decision maker takes an action other than replacing component i in decision epoch t. In this case, $P(c_{i,t+1} = w | c_{it} = w) = p_i$ where $0 < p_i < 1$ and $1 - p_i$ is the constant transition probability of failure for a working component i. When $a_t \in A \backslash \{rc_i\}$ and component i is non-working in period t, it will still be non-working in the subsequent period $t + 1$. So $P(c_{i,t+1} = nw | c_{it} = nw) = 1$. When $a_t = rc_i$, component i will be in its working state with probability one in the next period no matter whether it is working or not in the current period.

Table 1: Transition probability of component i

$a_t \in A_t \backslash \{rc_i\}$

c_{it}	$c_{i,t+1}$ w	$c_{i,t+1}$ nw
w	p_i	$1 - p_i$
nw	0	1

$a_t = rc_i$

c_{it}	$c_{i,t+1}$ w	$c_{i,t+1}$ nw
w	1	0
nw	1	0

3 Proposed Solution

The system under maintenance can be arbitrarily complex. There may be non-component nodes in the system which are not temporal and only transmits information from its parents to its children. After eliminating these nodes from the graph, we propose POMDPs as a solution to the maintenance problem with the reduced joint state space as the hidden process state space. This space grows exponentially with the number of component nodes in the graph. To overcome this difficulty we propose a hierarchical solution procedure where in the higher level we solve an aggregate model of the problem with an exact POMDP solver and in the lower level we disaggregate the aggregate solution using DBNs.

3.1 POMDP Formulation

The maintenance problem has a POMDP structure with the following exception: In a POMDP, there is usually a single variable defining the hidden process. However, in our problem the hidden process is more complex since it consists of several component nodes (Figure 1). One solution to this drawback is to merge all component nodes into a single mega node as in Figure 2. Here, the component nodes c_{1t},..c_{it},..,c_{It} are merged into the process node s_t. After the merge, the bold black arcs are added to the model, the grey arcs are deleted from the model and the rest of arcs are maintained as they are before the merge. Let S be the new process state space of s_t, T be the new set of transition probabilities of the process node, and O be the new set of observation probabilities. The following conversions should be done to formulate the problem as a POMDP. The new process state space S is the Cartesian product of the component state spaces and can be represented as $S = C_1 \times C_2 \times ... \times C_I$, where $|S| = \prod_{i=1}^{I} |C_i|$.

The new observation probabilities O can be constructed as follows:

$$P(o_t | s_t) = P(o_t | c_{1t}, c_{2t}, .., c_{It}), \qquad (1)$$

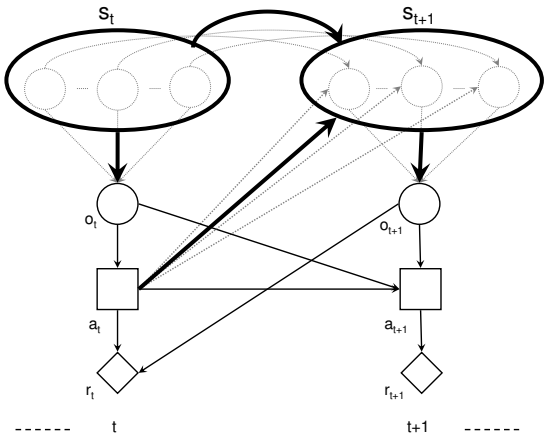


Figure 2: Part of the influence diagram after conversion to POMDP

which are in fact equal to the observation probabilities O' before merge. When components are independent as in Figure 2, the new transition probabilities T can be constructed from T_i' as in (2). For more general models where there are also dependencies among components, T can be constructed with (3), where $pa(c_{i,t+1})$ consists of all parents of $c_{i,t+1}$ including c_{it} and a_t.

$$P(s_{t+1}|s_t, a_t) = \prod_{i=1}^{I} P(c_{i,t+1}|c_{it}, a_t), \quad (2)$$

$$P(s_{t+1}|s_t, a_t) = \prod_{i=1}^{I} (c_{i,t+1}|pa(c_{i,t+1})). \quad (3)$$

3.2 Aggregate Model

We have formulated the maintenance problem as a POMDP with a hidden state space whose cardinality is $|S| = \prod_{i=1}^{I} |C_i|$. As i increases $|S|$ increases exponentially making the problem harder to solve. To overcome this difficulty, we can aggregate some states and form new aggregate states S_p. Let S^a be the new state space, built after aggregation, whose elements are S_p which are mutually exclusive and totally exhaustive sets (i.e., $\cup_{p=1}^{P} S_p = S,\ S_p \cap S_q = \emptyset,\ p \neq q\ p, q = 1..P$). Let P be the new cardinality of S^a, $|S^a| = P$. Let T^a and O^a be the new transition and observation probabilities of this aggregate model which are obtained from T and O respectively as follows:

$$P(s_{t+1} \in S_q | s_t \in S_p, a_t) = \frac{\sum_{s_{t+1} \in S_q} \sum_{s_t \in S_p} P(s_{t+1}|s_t, a_t)}{|S_p|}, \quad (4)$$

$$P(o_t | s_t \in S_p) = \frac{\sum_{s_t \in S_p} P(o_t|s_t)}{|S_p|}, \quad (5)$$

where the cardinality $|S_p|$ is a normalizing factor. When process states are aggregated, it is more meaningful to aggregate the related actions. Let A^a denote the new action space, built after aggregation, whose elements are A_j which are mutually exclusive and totally exhaustive sets (i.e., $\cup_{j=1}^{J} A_j = A,\ A_j \cap A_l = \emptyset,\ j \neq l\ j, l = 1..J$). Let J be the new cardinality of A^a, $|A^a| = J$.

When actions are aggregated, it affects the transition probabilities and the reward function since they depend on actions performed in each decision epoch. Let T^{aa} and R^a be the new transition probabilities and the new reward function of this aggregate model where actions are also aggregated in addition to states. T^{aa} and R^a are obtained from the original data, T and R, respectively as follows:

$$P(s_{t+1} \in S_q | s_t \in S_p, a_t \in A_j) = \frac{\sum_{a_t \in A_j} \sum_{s_{t+1} \in S_q} \sum_{s_t \in S_p} P(s_{t+1}|s_t, a_t)}{|A_j||S_p|}, \quad (6)$$

$$R(A_j, o) = \frac{\sum_{a \in A_j} R(a, o)}{|A_j|}, \quad (7)$$

where $|A_j||S_p|$ and $|A_j|$ are normalizing factors. In (6–7), actions are given equal probabilities to be performed during aggregation. However, when one prefers to replace some components more or less frequently than the others, giving weights to actions according to the desired frequency is more appropriate. Let W_a be the weight of action a. Let T_w^{aa} and R_w^a be the new transition probabilities and the new reward function of this weighted aggregate model where actions are also aggregated with respect to their weights in addition to the aggregation of states.

T_w^{aa} and R_w^a are obtained from the original data, T and R, respectively as follows:

$$P(s_{t+1} \in S_q | s_t \in S_p, a_t \in A_j) = \frac{\sum_{a_t \in A_j} \sum_{s_{t+1} \in S_q} \sum_{s_t \in S_p} W_a P(s_{t+1}|s_t, a_t)}{\sum_{a_t \in A_j} W_a |S_p|}, \quad (8)$$

$$R(A_j, o) = \frac{\sum_{a \in A_j} W_a R(a, o)}{\sum_{a \in A_j} W_a}. \quad (9)$$

Although some detailed information is lost after aggregation, the aggregate model has fewer process states and action states than the original model, which makes it easier to solve.

3.3 Disaggregation

The aggregate solution given by an exact POMDP solver is disaggregated to obtain an expected maintenance cost of the original problem. Disaggregation is performed by simulating the process with a DBN tool (Bayesian Network Toolbox) and applying troubleshooting approaches in the decision epoch, where replacement is planned in the aggregate policy, to obtain which component to replace.

3.3.1 DBN Formulation

The original problem is simulated with a DBN given in Figure 3, where action a_t is represented as a probabilistic node such that $a_t \in A$, $A = \{dn, rc_1, rc_2, ..., rc_I\}$. Solid arcs represent the causal relations between the components and the observation node. They constitute the conditional probabilities $P(o_t|c_{1t}..c_{It})$ for all t. The dashed arcs represent temporal relations of the components and actions between two consecutive time periods. They constitute the conditional probabilities $P(c_{i,t+1}|c_{it}, a_t)$. Temporal relations are the transition probabilities of components due to aging and replacement which are given in Table 1.

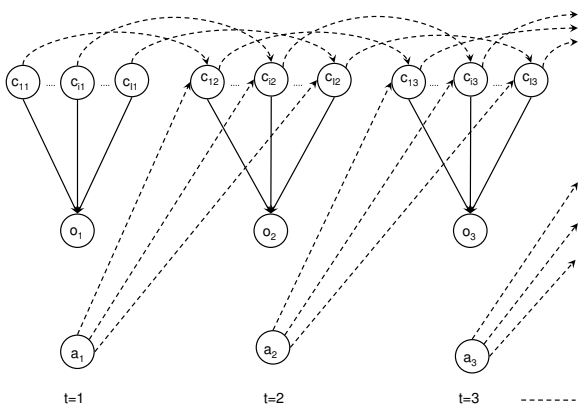


Figure 3: DBN representation of the disaggregate problem

3.3.2 Disaggregation Procedure

The disaggregation procedure takes the optimal policy of the aggregate POMDP model as input. This policy is an aggregate policy in terms of aggregate actions. It does not specify which component to replace when it decides to do a replacement. This is done by the disaggregation procedure. It is performed by simulating the original problem (before merge) with a DBN represented in Figure 3. Let ε be the evidence set containing the total evidence gathered so far. It is accumulated by two means in every period: one is the selected disaggregated action of the current period and the other is the sampled observation of the next period. In each period, if a replacement is required by the aggregate policy, the component which has the highest efficiency is selected. This constitutes the current disaggregated action. The efficiency measure is defined as follows:

$$ef_{it} = \frac{P(c_{i,t+1} = nw | \varepsilon \cup \{o_{t+1} = r\})}{\pi_i}, \quad (10)$$

where the numerator is the probability of component i being nonworking given the up-to-date evidence and also the condition that a red signal is observed in the next period although this may not be the case. Appending the latter one to the evidence indicates the component which explains observing a red signal in the next period. The denominator, π_i, is the replacement cost of component i and it makes the efficiency measure to trade off component's nonworking probability to its replacement cost. The component with the highest efficiency is selected to be replaced.

4 Computational Study

The design structure given in Table 2 is used in the computational study. Four factors are determined in the experimental design. Two of them are related with costs and the other two are related with probabilities in the problem. The values of each factor are seen in the related column of the table. When observation probabilities are different, probability of observing green differs as number of faults differs for in between states (other than the all working and all nonworking states). As an alternative case, invariant observation probabilities mean probability of observing a green signal is same for all intermediate states and it is the average of the observation probabilities of the different case. Each combination of factor values leads to a total of 32 data sets. After elimination of four symmetric data, we have a total of 28 data sets.

Table 2: Experimental design

Transition Working Probability	Replacement Cost	Red Cost	Observation Probability
Increasing Constant-low Constant-high Decreasing	Increasing Constant	High Low	Different Indifferent

For a four component dynamic system, when components are merged into a single process node, we have a total of $2^4 = 16$ states and 5 actions. We try to solve the problem with an exact POMDP solver, but it cannot be solved exactly. Then states are aggregated into three states such that S_1 is the all working state, S_3 is the all nonworking state, and S_2 is the in between state (at least one working and one nonworking component exist). Actions are aggregated into two actions such that A_1 is doing nothing and A_2 is replacing one of the components. Initially no weights are given to the actions in A_2 while aggregating. Observation and transition probabilities, and reward function are aggregated accordingly as in Equations 5, 6 and 7 respectively.

After aggregating the problem into three compound states and two compound actions, it can be solved with an exact POMDP solver (pomdp-solve) implementing the Witness algorithm (Cassandra et al., 1994) for a finite horizon of 100 periods. All data sets are also solved with an approximate POMDP solver, ZMDP software package (Smith, 2007), for a finite horizon of 100 periods which finds upper bounds for the cost function and gives POMDP policies as an output. The ZMDP policy file is also simulated with the DBN in Figure 3 to achieve a complete solution as in the case of the disaggregate solution. Both disaggregation and ZMDP simulation are replicated 50 times and the average cost of these is reported with their standard deviations in Table 3.

We observe that, in data sets d01-d04, d08, d10, d11, d15-d18, d22, d24 and d25, only the first component is replaced in all replacement times and replications. So, a better aggregation can be possible by giving more weights to the replacement actions of components which are replaced more frequently and giving no weights to the replacement actions whose components are not even replaced. So we re-aggregate the data sets having unequal transition probabilities or unequal replacement costs by giving the average number of replacements of each component as its weight to the corresponding replacement action. Data sets d06, d07, d13, d14, d20, d21, d27 and d28 have equal transition probabilities and replacement costs, so giving unequal weights do not lead to different aggregations other than the ones already constituted. The computational results of the weighted aggregation are also tabulated in Table 3.

The results show that average cost of most of the re-aggregated data sets improve (d01-d04, d08-d11, d15, d16, d22-d25) or remain the same (d05, d12, d19 and d26) except data sets d17 and d18 which have performed already well with the equal weighted aggregation. One can make new aggregations of replacement actions with the new average number of replacements of components. This goes on until a cycle of aggregate policies is determined, which means no new solutions will be available to the decision maker. However, there is no guarantee that disaggregation results will improve every time a new aggregation is performed as in the case of data sets d17 and d18.

Table 3: Computational results

Set	Equal Disagg	(std)	Weighted Disagg	(std)	ZMDP Sim.Avg	(std)	t-test p-value
d01	1497.3	(131.9)	**1220.8**	(103.6)	854.7	(129.2)	.0000 *
d02	1500.8	(72.27)	**1308**	(104.3)	1295.2	(85.96	.5046
d03	1068.4	(163.8)	**717.5**	(127)	605.52	(112.16)	.0000 *
d04	1218.8	(84.19)	**904.2**	(104.4)	915.86	(110.26)	.5871
d05	**1308.1**	(123.4)	1308.1	(123.4)	908.6	(96.46)	.0000 *
d06	**1479.4**	(85.93)	-	-	1495.2	(96.77)	.3901
d07	**955.4**	(130.8)	-	-	562.30	(81.95)	.0000 *
d08	414.6	(41.79)	**375.1**	(30.73)	348.74	(42.23)	.0006 *
d09	496.2	(4.11)	**398.4**	(22.22)	396.6	(24.65)	.7021
d10	296.6	(36.54)	**269.7**	(44.09)	232.6	(40.46)	.0000 *
d11	324.9	(25.54)	**299.4**	(30.49)	292.38	(33.35)	.2747
d12	**400.8**	(43.16)	400.8	(43.16)	476.2	(.57)	.0000 +
d13	**554.4**	(23.12)	-	-	495.6	(4.70)	.0000 *
d14	**280.3**	(31.87)	-	-	227.7	(40.17)	.0000 *
d15	1336.2	(126.1)	**886.8**	(95.33)	851.3	(102.4)	.0761
d16	1259.7	(87.37)	**878.4**	(90.25)	898.8	(125.3)	.3525
d17	**853.26**	(105.8)	924.76	(125.9)	764.66	(115.60)	.0001 *
d18	**873.32**	(86.54)	1150.2	(105.87)	892.7	(97.9)	.2958
d19	**1059.8**	(123.2)	1059.8	(123.2)	1120.6	(102.2)	.0085 +
d20	**1298.6**	(103.4)	-	-	1306.4	(91.65)	.6906
d21	**853.9**	(119.4)	-	-	690.7	(105.09)	.0000 *
d22	381.38	(44.52)	**315.28**	(26.8)	377.1	(36.5)	.0000 +
d23	496.2	(4.58)	**295.7**	(21.75)	303.7	(21.3)	.0661
d24	238.32	(31.17)	**221.3**	(25.24)	242.12	(29.18)	.0002 +
d25	245.16	(25.35)	**236.8**	(23.82)	234.94	(20.85)	.6788
d26	**341**	(47.78)	341	(47.78)	274.4	(36.6)	.0000 *
d27	**501.1**	(23.28)	-	-	494.9	(5.1)	.0689
d28	**242.3**	(33.06)	-	-	236.7	(25.88)	.3479

It is of interest to analyze the performance of the two procedures in each data set in order to understand what type of data sets our procedure is successful at. So, for each data set, we perform hypothesis test for two samples and use two-sided t-test to test whether the means of simulated ZMDP cost and the best disaggregation cost (with bold number) are equal or not in each data set. The resulting p-value is given in the table for each data set. When significance level is taken as 0.05, data sets with p-value less than 0.05 are marked in the table with * or + indicating that the simulated ZMDP cost is significantly less than the disaggregation cost or vice versa, respectively. There are 15 data sets out of 28 where the two methods significantly differ in terms of cost values. 11 of these data sets belong to the case where simulated ZMDP cost is significantly lower than disaggregate cost whereas 4 of them belong to the case where disaggregate cost is significantly lower. If we look at the distribution of 11 data sets where ZMDP cost is significantly lower than disaggregate cost, we will see that 8 of them lie within the data sets with different observation probabilities for the in between states (d01-d14). Alternatively only 1 out of 4 data sets where disaggregate cost is significantly lower than simulated ZMDP cost, lies in d01-d14. These results give insight that our solution procedure is more successful at the data sets whose observation probabilities of the aggregated states are invariant.

One may be interested in obtaining the overall performance of the proposed procedure by using one-sided paired t-test. Experimental results already show that there are plenty of instances where our procedure is outperformed by the simulated ZMDP. However simulated ZMDP has two drawbacks: First, in order to run ZMDP in a finite horizon, time information is added explicitly into the state information. So, when the number of components increase, state space and hence size of the ZMDP input file grows exponentially. Second, when ZMDP is run, it gives an upper bound for the cost function. However to obtain full solutions as in the case of our procedure, we also simulate the ZMDP output file. This requires more computational time than simulating the aggregate model of our procedure.

5 Conclusion

We tackle the maintenance problem of a complex system where system components are partially observable via indirect signals, and present a hierarchical solution procedure to solve it. The complex POMDP problem is aggregated in terms of states and actions such that it can be solved exactly and the optimal policy is implemented on the system by simulating it with DBNs. Our solutions are compared with the solutions of the approximate POMDP solver, which uses full information state space, on 28 data sets. The results show that when observation probabilities are invariant among the states aggregated into the same compound state, our procedure performs better.

In the aggregation, we prefer to aggregate the states into three compound states which are all working, all nonworking and in between states; and the actions into two compound actions which are doing nothing and doing a replacement. A POMDP with three states and two actions can always be solved exactly. However, significant probabilistic information can be lost. The disaggregation result can be at most as good as the quality of the aggregate policy. The action aggregation can be improved by giving weights to replacement actions while the state aggregation is harder to improve. The average number of replacements of components obtained from the disaggregation solution can be used as weights, however determining the appropriate weights for action aggregation from the parameters of the problem will be a better way since this will reduce the number of aggregations and hence the effort. This can be a future study.

References

A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla. 2001. Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. *Reliability Engineering and System Safety*, 71:249–260.

Anthony R. Cassandra, Leslie P. Kaelband, and Michael L. Littman. 1994. Acting optimally in partially observable stochastic domains. Technical Report CS-94-20, Brown University, Providence, Rhode Island.

Milos Hauskrecht. 2000. Value function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94.

David Heckerman, John S. Breese, and Koos Rommelse. 1995. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57.

Finn V. Jensen, Uffe Kjærulff, Brian Kristiansen, Helge Langseth, Claus Skaanning, Jiri Vomlel, and Marta Vomlelováá. 2001. The SACSO methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 15(4):321–333.

Jayant Kalagnam and Max Henrion. 1988. A comparison of decision analysis and expert rules for sequential analysis. In *Proceedings of 4th Conference on Uncertainty in Artificial Intelligence*, pages 271–281.

Helge Langseth and Luigi Portinale. 2006. Bayesian networks in reliability. *Reliability Engineering and System Safety*, 92:92–108.

Alexandre Muller, Philippe Weber, and A. Ben Salem. 2004. Process model-based dynamic Bayesian networks for prognostic. In *Proceedings of 4th International Conference on Intelligent Systems Design and Applications*, pages 849–854.

Donald Rosenfield. 1976. Markovian deterioration with uncertain information. *Operations Research*, 24(1):141–155.

Sheldon M. Ross. 1971. Quality control under Markovian deterioration. *Management Science*, 17(9):587–596.

Trey Smith. 2007. ZMDP software for POMDP and MDP planning. http://www.cs.cmu.edu/ trey/zmdp/README.

José Gerardo Torres-Toledano and Luis Enrique Sucar. 1998. Bayesian networks for reliability analysis of complex systems. In *Proceedings of 6th Ibero-American Conference on AI: Progress in Artificial Intelligence*, pages 195–206.

Philippe Weber and Lionel Jouffe. 2003. Reliability modeling with dynamic Bayesian networks. In *Proceedings of 5th IFAC Symposium SAFEPROCESS'03*, pages 57–62.

Robert L. Welch and Travis V. Thelen. 2000. Dynamic reliability analysis in an operational context: the Bayesian network perspective. In *Proceedings of Dynamic Reliability: Future Directions*, pages 277–307.

Reading Dependencies from Polytree-Like Bayesian Networks Revisited

Jose M. Peña
ADIT, IDA, Linköping University, Sweden

Abstract

We present a graphical criterion for reading dependencies from the minimal directed independence map G of a graphoid p, under the assumption that G is a polytree and p satisfies weak transitivity. We prove that the criterion is sound and complete. We argue that assuming weak transitivity is not too restrictive.

1 Introduction

A minimal directed independence map G of an independence model p is typically used to read independencies holding in p. However, G can also be used to read dependencies holding in p. For instance, if p is a graphoid that is faithful to G, then lack of vertex separation is a sound and complete graphical criterion for reading dependencies from G. If p is simply a graphoid, then there also exists a sound and complete graphical criterion for reading dependencies from G (Bouckaert, 1995). In (Peña, 2007), we present a further sound and complete graphical criterion for reading dependencies from G under the assumption that G is a polytree and p is a graphoid that satisfies composition and weak transitivity. In this paper, we revisit the latter work and drop the assumption that p satisfies composition. In general, the more assumptions a criterion makes about G and p the more powerful it is (i.e. the more dependencies it can read from G) but the less applicable it is (i.e. the smaller the set of independence models it can be applied to). Then, our new criterion may be seen as being in between the criteria in (Bouckaert, 1995) and (Peña, 2007): It is more (resp. less) powerful but less (resp. more) applicable than the former (resp. latter) criterion. See Section 5 for an example.

The rest of the paper is organized as follows. Section 2 is devoted to the preliminaries, Section 3 to our assumptions, Section 4 to our contribution, and Section 5 to the discussion.

2 Preliminaries

Let U denote a set of random variables. The elements of U are not distinguished from singletons, and the union of the sets $U_1, \ldots, U_n \subseteq U$ is written as the juxtaposition $U_1 \ldots U_n$. When evaluating an expression, the union of sets precedes the set difference. Let X, Y, Z and W denote four mutually disjoint subsets of U. An independence model p is a set of independencies of the form X is independent of Y given Z. We denote that an independence is in p by $X \perp_p Y|Z$ and that an independence is not in p by $X \not\perp_p Y|Z$. In the latter case, we say that the dependence $X \not\perp_p Y|Z$ is in p. An independence model is a graphoid if it satisfies the following properties: Symmetry $X \perp_p Y|Z \Rightarrow Y \perp_p X|Z$, decomposition $X \perp_p YW|Z \Rightarrow X \perp_p Y|Z$, weak union $X \perp_p YW|Z \Rightarrow X \perp_p Y|ZW$, contraction $X \perp_p Y|ZW \wedge X \perp_p W|Z \Rightarrow X \perp_p YW|Z$, and intersection $X \perp_p Y|ZW \wedge X \perp_p W|ZY \Rightarrow X \perp_p YW|Z$.

We say that a node C is a collider in a route in a directed and acyclic graph (DAG) if $A \rightarrow C \leftarrow B$ is a subroute of the route. Note that A and B may coincide since we are dealing with a route and not with a path. A route in a DAG is said to be superactive wrt Z when (i) every collider node in the route is in Z, and (ii) every non-collider node in the route is outside Z. When there is no route in a DAG G between a node in X and a node in Y that is superactive wrt Z, we say that X is separated from Y given Z in G and denote it as $X \perp_G Y|Z$.

Pp. 225–233 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

This definition of separation in DAGs is equivalent to other more common definitions (Studený, 1998). Given an undirected graph (UG) G, we say that X is separated from Y given Z in G and denote it as $X \perp_G Y|Z$ when every path in G between a node in X and a node in Y contains a node in Z. An independence model p is faithful to an UG or DAG G when $X \perp_p Y|Z$ iff $X \perp_G Y|Z$. A DAG G is a directed independence map of an independence model p when $X \perp_p Y|Z$ if $X \perp_G Y|Z$. Moreover, G is a minimal directed independence (MDI) map of p when removing any edge from G makes it cease to be an independence map of p. If G is a MDI map of p, then the parents of a node A in G, $Pa(A)$, are the smallest subset of the nodes preceding A in a given total ordering of U, $Pre(A)$, such that $A \perp_p Pre(A) \setminus Pa(A)|Pa(A)$. We denote the children of A in G by $Ch(A)$. Finally, recall that a polytree is a directed graph without undirected cycles.

3 WT Graphoids

Let X, Y and Z denote three mutually disjoint subsets of U. Let $V \in U \setminus XYZ$. We call WT graphoid to any graphoid p that satisfies weak transitivity $X \perp_p Y|Z \wedge X \perp_p Y|ZV \Rightarrow X \perp_p V|Z \vee V \perp_p Y|Z$. This paper studies WT graphoids. We regard WT graphoids as worth studying because important families of probability distributions are WT graphoids. For instance, any probability distribution that is Gaussian or faithful to some UG or DAG is a WT graphoid (Pearl, 1988). The following theorem implies that there also exist probability distributions that are WT graphoids although they are neither Gaussian nor faithful to any UG or DAG. See (Peña et al., 2009) for the proof and examples.

Theorem 1. *Let p be a probability distribution that is a WT graphoid and let $W \subseteq U$. Then, $p(U \setminus W)$ is a WT graphoid. If $p(U \setminus W|W = w)$ has the same independencies for all value w of W, then $p(U \setminus W|W = w)$ for any w is a WT graphoid.*

The following theorem introduces a new property that every WT graphoid satisfies.

Theorem 2. *Let p be a WT graphoid. Then, p satisfies the following property: Intersectional weak transitivity $X \perp_p Y|Z \wedge X \perp_p Y|ZV \Rightarrow X \perp_p V|ZY \vee V \perp_p Y|ZX$.*

Proof. Assume to the contrary that $X \not\perp_p V|ZY$ and $V \not\perp_p Y|ZX$. Then,

1. $X \not\perp_p VY|Z$ and $VX \not\perp_p Y|Z$ by the contrapositive form of weak union on $X \not\perp_p V|ZY$ and $V \not\perp_p Y|ZX$
2. $X \not\perp_p V|Z$ and $V \not\perp_p Y|Z$ by the contrapositive form of contraction on (1) and $X \perp_p Y|ZV$
3. $X \not\perp_p Y|ZV$ by the contrapositive form of weak transitivity on (2) and $X \perp_p Y|Z$.

However, (3) contradicts the antecedent of the property. □

4 Reading Dependencies

If G is a MDI map of a WT graphoid p then we know, by construction of G, that $A(Pre(B) \setminus Pa(B)) \not\perp_p B|Pa(B) \setminus A$ for all the edges $A \rightarrow B$ in G. We call these dependencies the dependence base of p for G. Further dependencies in p can be derived from the dependence base via the WT graphoid properties. For this purpose, we rephrase the WT graphoid properties in their contrapositive form as follows. Symmetry $Y \not\perp_p X|Z \Rightarrow X \not\perp_p Y|Z$. Decomposition $X \not\perp_p Y|Z \Rightarrow X \not\perp_p YW|Z$. Weak union $X \not\perp_p Y|ZW \Rightarrow X \not\perp_p YW|Z$. Contraction $X \not\perp_p YW|Z \Rightarrow X \not\perp_p Y|ZW \vee X \not\perp_p W|Z$ is problematic for deriving new dependencies because it contains a disjunction in the consequent and, thus, we split it into two properties: Contraction1 $X \not\perp_p YW|Z \wedge X \perp_p Y|ZW \Rightarrow X \not\perp_p W|Z$, and contraction2 $X \not\perp_p YW|Z \wedge X \perp_p W|Z \Rightarrow X \not\perp_p Y|ZW$. Likewise, intersection gives rise to intersection1 $X \not\perp_p YW|Z \wedge X \perp_p Y|ZW \Rightarrow X \not\perp_p W|ZY$, and intersection2 $X \not\perp_p YW|Z \wedge X \perp_p W|ZY \Rightarrow X \not\perp_p Y|ZW$. Note that intersection1 and intersection2 are equivalent and, thus, we refer to them simply as intersection. Similarly, weak transitivity gives rise to weak transitivity1 $X \not\perp_p V|Z \wedge V \not\perp_p Y|Z \wedge X \perp_p Y|Z \Rightarrow X \not\perp_p Y|ZV$, and weak transitivity2 $X \not\perp_p V|Z \wedge V \not\perp_p Y|Z \wedge X \perp_p Y|ZV \Rightarrow X \not\perp_p Y|Z$. Finally, intersectional weak transitivity gives rise to intersectional weak transitivity1 $X \not\perp_p V|ZY \wedge V \not\perp$

$_pY|ZX \wedge X \perp_p Y|Z \Rightarrow X \not\perp_p Y|ZV$, and intersectional weak transitivity2 $X \not\perp_p V|ZY \wedge V \not\perp_p Y|ZX \wedge X \perp_p Y|ZV \Rightarrow X \not\perp_p Y|Z$. The independence in the antecedent of any of the properties above holds if the corresponding separation statement holds in G. This is the best solution we can hope for because separation is sound and complete. Separation is sound in the sense that it only identifies independencies in p. Moreover, separation is complete in the sense that it identifies all the independencies in p that can be identified by studying G alone (Peña, 2007). We call the WT (resp. IWT) graphoid closure of the dependence base of p for G to the set of dependencies that are in the dependence base of p for G plus those that can be derived from it by applying the first eight (resp. all the ten) properties above. The following example shows that the WT and IWT graphoid closures of a dependence base do not coincide in general.

Example 1. Let p be a probability distribution over $U = \{A, B, C\}$ where A, B and C are binary random variables. Let $p(A, B)$ be uniform and $C = XOR(A, B)$. Note that $A \perp_p B$, $A \perp_p C$ and $B \perp_p C$ are the only independencies in p. Then, p is a WT graphoid. Let G denote the DAG $A \rightarrow C \leftarrow B$. Note that G is a MDI map of p. Now, note that $A \not\perp_p B|C$ is in the IWT graphoid closure of the dependence base of p for G: The dependence base of p for G is $\{A \not\perp_p C|B, B \not\perp_p C|A\}$, which implies $A \not\perp_p B|C$ by intersectional weak transitivity1 and $A \perp_G B$. However, the WT graphoid closure of the dependence base of p for G is $\{A \not\perp_p C|B, B \not\perp_p C|A, A \not\perp_p BC, AB \not\perp_p C, B \not\perp_p AC, C \not\perp_p A|B, C \not\perp_p B|A, BC \not\perp_p A, C \not\perp_p AB, AC \not\perp_p B\}$ which does not contain $A \not\perp_p B|C$.

Hereinafter, we use $A : B$ to denote a route between two nodes A and B in a DAG G. We also use $A : B$ to denote the nodes in the route. It should be clear from the context which of the two meanings is being used. We define the parents of a route $A : B$ as $Pa(A : B) = [\cup_{C \rightarrow D \in A:B} Pa(D)] \setminus (A : B)$. We say that a route $A : B$ is minimally superactive wrt X, Y and Z in G if (i) $A \in X$ and $B \in Y$, (ii) $A : B$ is superactive wrt Z, and (iii) no proper subroute of $A : B$ is minimally superactive wrt X, Y and Z in G. Finally, we introduce our graphical criterion for reading dependencies from a polytree-like MDI map of a WT graphoid.

Definition 1. Let G be a polytree. Let X, Y and Z denote three mutually disjoint subsets of U. We say that $X \sim_G Y|Z$ holds if

- there exist two nodes $A \in X$ and $B \in Y$ and a single route $A : B$ between them that is minimally superactive wrt X, Y and Z in G, and
- for all $A' \in Pa(A : B)$, $A' \in XYZ \setminus AB$ or $A' \sim_{G_{\setminus A'}} XYZ \setminus AB$ where $G_{\setminus A'}$ is the DAG resulting from removing from G the edge between A' and its child in $A : B$.

Note the recursive flavor of the definition above: A base case ($A' \in XYZ \setminus AB$) and a recursive call ($A' \sim_{G_{\setminus A'}} XYZ \setminus AB$). The next theorem proves that the criterion in Definition 1 is sound. We prove first some auxiliary lemmas.

Lemma 1. *Let G be a polytree-like MDI map of a WT graphoid p. Let A and B be two nodes such that $A \sim_G B|Z$ holds due to a route $A : B$ with no collider node. Let $Pa(A : B) \subseteq Z$. Then, $A \not\perp_p B|Z$.*

Proof. We prove the lemma by induction over the length of $A : B$. We first prove the lemma for length one, i.e. $A : B$ is $A \rightarrow B$ or $A \leftarrow B$. Assume without loss of generality that $A : B$ is $A \rightarrow B$. Let Z^A denote the nodes in Z that are in $Pa(A)$ or connected to A by an undirected path that passes through $Pa(A)$. Let Z_A denote the nodes in Z that are in $Ch(A) \setminus B$ or connected to A by an undirected path that passes through $Ch(A) \setminus B$. Let Z^B denote the nodes in Z that are in $Pa(B) \setminus A$ or connected to B by an undirected path that passes through $Pa(B) \setminus A$. Note that $Pa(B) \setminus A \subseteq Z^B$ because we have assumed that $Pa(A : B) \subseteq Z$. Then,

1. $A(Pre(B) \setminus Pa(B)) \not\perp_p B|Pa(B) \setminus A$ from the dependence base of p for G
2. $A \not\perp_p B|Pa(B) \setminus A$ by contraction1 on (1) and $Pre(B) \setminus Pa(B) \perp_G B|(Pa(B) \setminus A)A$
3. $AZ^A Z_A \not\perp_p B|Pa(B) \setminus A$ by decomposition on (2)
4. $A \not\perp_p B|(Pa(B) \setminus A)Z^A Z_A$ by intersection on (3) and $Z^A Z_A \perp_G B|(Pa(B) \setminus A)A$

5. $A \not\perp_p B(Z^B \setminus (Pa(B) \setminus A))|(Pa(B) \setminus A)Z^A Z_A$ by decomposition on (4)

6. $A \not\perp_p B|Z^A Z_A Z^B$ by contraction2 on (5) and $A \perp_G Z^B \setminus (Pa(B) \setminus A)|(Pa(B) \setminus A)Z^A Z_A$

7. $A \not\perp_p B(Z \setminus Z^A Z_A Z^B)|Z^A Z_A Z^B$ by decomposition on (6)

8. $A \not\perp_p B|Z$ by intersection on (7) and $A \perp_G Z \setminus Z^A Z_A Z^B|Z^A Z_A Z^B B$.

Assume as induction hypothesis that the lemma holds when the length of $A : B$ is smaller than l. We now prove the lemma for length l. Let C be any node in $A : B$ except A and B. Recall that $A : B$ has no collider node. Thus, $C \notin Z$ because $A : B$ is minimally superactive wrt A, B and Z in G. Moreover, $C \notin Z$ implies $A \perp_G B|ZC$, $A \sim_G C|Z$, and $B \sim_G C|Z$. The latter two statements imply $A \not\perp_p C|Z$ and $B \not\perp_p C|Z$ by the induction hypothesis, which together with $A \perp_G B|ZC$ imply $A \not\perp_p B|Z$ by weak transitivity2. □

Lemma 2. *Let G be a polytree-like MDI map of a WT graphoid p. Let A and B be two nodes such that $A \sim_G B|Z$ holds due to a route $A : B$ of the form $A \to C \to \ldots \to D \leftarrow \ldots \leftarrow C \leftarrow B$ with possibly $C = D$. Let $Pa(A : B) \subseteq Z$. Then, $A \not\perp_p B|Z$.*

Proof. Let Z_D be the descendants of D that are in Z. Note that $A \perp_G B|Z \setminus Z_D D$ as $A : B$ is the only route between A and B that is minimally superactive wrt A, B and Z in G. Then,

1. $A \not\perp_p D|(Z \setminus Z_D D)B$ by Lemma 1

2. $B \not\perp_p D|(Z \setminus Z_D D)A$ by Lemma 1

3. $A \not\perp_p B|(Z \setminus Z_D D)D$ by intersectional weak transitivity1 on (1), (2), and $A \perp_G B|Z \setminus Z_D D$

4. $A \not\perp_p B Z_D|(Z \setminus Z_D D)D$ by decomposition on (3)

5. $A \not\perp_p B|Z$ by contraction2 on (4) and $A \perp_G Z_D|(Z \setminus Z_D D)D$. □

Lemma 3. *Let G be a polytree-like MDI map of a WT graphoid p. Let $X \sim_G Y|Z$ hold due to a route $A : B$ with $A \in X$ and $B \in Y$. Let $Pa(A : B) \subseteq XYZ \setminus AB$. Then, $X \not\perp_p Y|Z$.*

Proof. Let $W = XYZ \setminus AB$. Let $D_1, \ldots, D_n$ be the collider nodes in $A : B$. Then, for all i, $A : B$ has a subroute of the form $A_i \to C_i \to \ldots \to D_i \leftarrow \ldots \leftarrow C_i \leftarrow B_i$ with $A_i \neq B_i$ but possibly $C_i = D_i$. Let W_{D_i} denote the descendants of D_i that are in W. Let W_{C_i} denote the descendants of C_i that are in $W \setminus W_{D_i} D_i$. Let $W' = \cup_i W_{C_i}$. We first prove $A \not\perp_p B|W \setminus W'$. Note that $Pa(A : B) \subseteq W \setminus W'$ and, thus, that $A \sim_G B|W \setminus W'$ holds due to $A : B$. Then, we can divide $A : B$ into subroutes such that each of them is of the form of the route in Lemma 1 or 2. We prove $A \not\perp_p B|W \setminus W'$ by induction over the number of such subroutes. If the number of subroutes is one, then the result is immediate by Lemma 1 or 2. Assume as induction hypothesis that the result holds when the number of subroutes is smaller than l. We now prove the result when this number is l. Let E be any node in $A : B$ where two of the subroutes meet. Note that E is a non-collider node in $A : B$ and, thus, $E \notin W \setminus W'$. Moreover, $E \notin W \setminus W'$ implies $A \perp_G B|(W \setminus W')E$, $A \sim_G E|W \setminus W'$, and $E \sim_G B|W \setminus W'$. Then,

1. $A \not\perp_p E|W \setminus W'$ and $E \not\perp_p B|W \setminus W'$ by $A \sim_G E|W \setminus W'$, $E \sim_G B|W \setminus W'$ and the induction hypothesis

2. $A \not\perp_p B|W \setminus W'$ by weak transitivity2 on (1) and $A \perp_G B|(W \setminus W')E$.

Finally, let $X' \subseteq X \setminus A$ and $Y' \subseteq Y \setminus B$ contain the nodes in W' that are not descendant of another node in X' or Y'. Note that X' and Y' must exist for $A : B$ to be the only route between A and B that is minimally superactive wrt X, Y and Z in G. Let $W_{X'}$ (resp. $W_{Y'}$) contain the descendants of X' (resp. Y') that are in W'. Then,

3. $AX'W_{X'} \not\perp_p B|W \setminus W'$ by decomposition on (2)

4. $AX' \not\perp_p B|(W \setminus W')W_{X'}$ by intersection on (3) and $W_{X'} \perp_G B|(W \setminus W')AX'$

5. $AX' \not\perp_p BY'W_{Y'}|(W \setminus W')W_{X'}$ by decomposition on (4)

6. $AX' \not\perp_p BY'|(W \setminus W')W_{X'}W_{Y'}$ by intersection on (5) and $AX' \perp_G W_{Y'}|(W \setminus W')W_{X'}BY'$

7. $X \not\perp_p Y|Z$ by decomposition and weak union on (6). □

If a route $A : B$ in a DAG has a subroute of the form $C \to D \to \ldots \to E \leftarrow \ldots \leftarrow D \leftarrow F$ with $C \neq F$ but possibly $D = E$, the subroute $D \to \ldots \to E \leftarrow \ldots \leftarrow D$ is called a rope.

Theorem 3. *Let G be a polytree-like MDI map of a WT graphoid p. If $X \sim_G Y|Z$, then $X \not\perp_p Y|Z$ is in the IWT graphoid closure of the dependence base of p for G.*

Proof. Let $X \sim_G Y|Z$ hold due to a route $A : B$ with $A \in X$ and $B \in Y$. Let $W = XYZ \setminus AB$. We prove the theorem by induction over the total number of recursive calls performed by $X \sim_G Y|Z$. If this number is zero, then the theorem is immediate by Lemma 3. Assume as induction hypothesis that the theorem holds when the total number of recursive calls is smaller than l. We now prove the theorem when this number is l. Let $A' \sim_{G_{\setminus A'}} W$ with $A' \in Pa(E)$ for some $E \in A : B$ be any recursive call performed by $X \sim_G Y|Z$. Let $A' \sim_{G_{\setminus A'}} W$ hold due to a route $A' : B'$ with $B' \in W$. We consider two scenarios. The first scenario is when E is outside every rope in $A : B$. Then, $A : B$ must have a subroute of the form $C \rightarrow E$ or $E \leftarrow D$ for the recursive call $A' \sim_{G_{\setminus A'}} W$ to be performed. Assume without loss of generality that the subroute is of the form $C \rightarrow E$. Let W_E denote the descendants of E that are in W. Note that $X \sim_G Y|Z$ implies that $A \sim_G A'|(W \setminus W_E B')E$ holds. To see it, note that the subroute of $A : B$ between A and E followed by $E \leftarrow A'$, here denoted $A : A'$, is the only route between A and A' that is minimally superactive wrt A, A' and $(W \setminus W_E B')E$ in G. Moreover, every recursive call that $A \sim_G A'|(W \setminus W_E B')E$ performs is of the form $A'' \sim_{G_{\setminus A''}} (W \setminus W_E B')E$ with $A'' \in Pa(A : A')$. This recursive call holds because $A'' \in Pa(A : B)$ and, thus, $X \sim_G Y|Z$ performs the recursive call $A'' \sim_{G_{\setminus A''}} W$ and W_E, B' and E are not used in it. By a similar reasoning, one can prove that $A' \sim_{G_{\setminus A'}} W$ implies that $A' \sim_G B'|(W \setminus W_E B')E$ holds. Then,

1. $A \not\perp_p A'|(W \setminus W_E B')E$ by $A \sim_G A'|(W \setminus W_E B')E$ and the induction hypothesis
2. $A' \not\perp_p B'|(W \setminus W_E B')E$ by $A' \sim_G B'|(W \setminus W_E B')E$ and the induction hypothesis
3. $A \not\perp_p B'|(W \setminus W_E B')E$ by weak transitivity2 on (1), (2), and $A \perp_G B'|(W \setminus W_E B')EA'$
4. $A \not\perp_p B'E|W \setminus W_E B'$ by weak union on (3)
5. $A \not\perp_p E|W \setminus W_E$ by contraction2 on (4) and $A \perp_G B'|W \setminus W_E B'$
6. $A \not\perp_p EW_E|W \setminus W_E$ by decomposition on (5)
7. $A \not\perp_p E|W$ by intersection on (6) and $A \perp_p W_E|(W \setminus W_E)E$
8. $E \not\perp_p B|W$ by $E \sim_G B|W$ and the induction hypothesis
9. $A \not\perp_p B|W$ by weak transitivity2 on (7), (8), and $A \perp_G B|WE$
10. $X \not\perp_p Y|Z$ by weak union and decomposition on (9).

The second scenario that we consider in the proof is when E is in a rope in $A : B$. In this case, $A : B$ has a subroute of the form $C \rightarrow D \rightarrow \ldots \rightarrow E \rightarrow \ldots \rightarrow F \leftarrow \ldots \leftarrow E \leftarrow \ldots \leftarrow D \leftarrow H$ with $C \neq H$ but possibly $D = E$ and/or $E = F$. Let W_F denote the descendants of F that are in W. Let W' be as in the proof of Lemma 3. Note that $C \perp_G H|W \setminus W'W_F F$ because $A : B$ is the only route between A and B that is minimally superactive wrt X, Y and Z in G. Then,

11. $C \not\perp_p F|(W \setminus W'W_F F)H$ by considering the first scenario for $C \sim_G F|(W \setminus W'W_F F)H$
12. $H \not\perp_p F|(W \setminus W'W_F F)C$ by considering the first scenario for $H \sim_G F|(W \setminus W'W_F F)C$
13. $C \not\perp_p H|(W \setminus W'W_F F)F$ by intersectional weak transitivity1 on (11), (12), and $C \perp_G H|W \setminus W'W_F F$
14. $C \not\perp_p HW_F|(W \setminus W'W_F F)F$ by decomposition on (13)
15. $C \not\perp_p H|W \setminus W'$ by contraction2 on (14) and $C \perp_G W_F|(W \setminus W'W_F F)F$
16. $A \not\perp_p C|W \setminus W'$ by $A \sim_G C|W \setminus W'$ and the induction hypothesis
17. $A \not\perp_p H|W \setminus W'$ by weak transitivity2 on (15), (16), and $A \perp_G H|(W \setminus W')C$
18. $H \not\perp_p B|W \setminus W'$ by $H \sim_G B|W \setminus W'$ and the induction hypothesis
19. $A \not\perp_p B|W \setminus W'$ by weak transitivity2 on (17), (18), and $A \perp_G B|(W \setminus W')H$
20. $X \not\perp_p Y|Z$ follows from (19) by repeating the steps (3)-(7) in the proof of Lemma 3.

Finally, note that we have derived (20) from the dependence base of p for G by using only the ten properties introduced at the beginning of Section 4. Thus, $X \not\perp_p Y|Z$ is in the IWT graphoid closure of the dependence base of p for G. □

The theorem below proves that the criterion in Definition 1 is complete in certain sense.

Theorem 4. *Let G be a polytree-like MDI map of a WT graphoid p. If $X \not\perp_p Y|Z$ is in the IWT graphoid closure of the dependence base of p for G, then $X \sim_G Y|Z$.*

Proof. Clearly, all the dependencies in the dependence base of p for G are identified by the criterion in Definition 1. It only remains to prove that the criterion satisfies the ten properties introduced at the beginning of Section 4.

• Symmetry $Y \sim_G X|Z \Rightarrow X \sim_G Y|Z$.

Trivial.

• Weak union $X \sim_G Y|ZW \Rightarrow X \sim_G YW|Z$.

We prove a simplified version of the property: We assume that W contains a single node. Repeated application of this simplified property proves the original property. Let $X \sim_G Y|ZW$ hold due to a route $A : B$ with $A \in X$ and $B \in Y$. We prove the simplified property by induction over the number of collider nodes in $A : B$. If this number is zero, then the proof is immediate because W cannot be in $A : B$ for $A : B$ to be minimally superactive wrt X, Y and ZW in G. Assume as induction hypothesis that the simplified property holds when the number of collider nodes in $A : B$ is smaller than l. We now prove the simplified property when this number is l. The proof is immediate unless W is in $A : B$. If the latter occurs, then $X \sim_G YW|Z$ holds due to $A : W$, i.e. the subroute of $A : B$ between A and W. To see it, note that $A : W$ is the only route between A and W that is minimally superactive wrt X, YW and Z in G. Note also that W must be a collider node in $A : B$ for $A : B$ to be minimally superactive wrt X, Y and ZW in G. Thus, $A : B$ has a subroute of the form $C \rightarrow D \rightarrow \ldots \rightarrow W \leftarrow \ldots \leftarrow D \leftarrow E$ with $C \neq E$ but possibly $D = W$. Then, every recursive call that $X \sim_G YW|Z$ performs belongs to one of the following two groups. The first group consists of the recursive calls $A' \sim_{G_{\setminus A'}} XYZW \setminus AW$ with $A' \in Pa(A : W) \setminus E$. These recursive calls hold because $A' \in Pa(A : B)$ and, thus, $X \sim_G Y|ZW$ performs the recursive calls $A' \sim_{G_{\setminus A'}} XYZW \setminus AB$ and B and W are not used in them. The second group consists of the recursive call $E \sim_{G_{\setminus E}} XYZW \setminus AW$. We prove that $E \sim_{G_{\setminus E}} Y|XZW \setminus A$ holds, which implies that $E \sim_{G_{\setminus E}} XYZW \setminus A$ holds by repeated application of the induction hypothesis and, since W is not used in the recursive call, $E \sim_{G_{\setminus E}} XYZW \setminus AW$ holds too. To see it, note that the subroute of $A : B$ between E and B, here denoted $E : B$, is the only route between E and B that is minimally superactive wrt E, Y and $XZW \setminus A$ in $G_{\setminus E}$. Moreover, every recursive call that $E \sim_{G_{\setminus E}} Y|XZW \setminus A$ performs is of the form $A' \sim_{(G_{\setminus E})_{\setminus A'}} XYZW \setminus AB$ with $A' \in Pa(E : B)$. This recursive call holds because $A' \in Pa(A : B)$ and, thus, $X \sim_G Y|ZW$ performs the recursive call $A' \sim_{G_{\setminus A'}} XYZW \setminus AB$.

• Decomposition $X \sim_G Y|Z \Rightarrow X \sim_G YW|Z$.

We prove a simplified version of the property: We assume that W contains a single node. Repeated application of this simplified property proves the original property. Let $X \sim_G Y|Z$ hold due to a route $A : B$ with $A \in X$ and $B \in Y$. The proof is immediate unless W is in $A : B$. If the latter occurs, then, $X \sim_G YW|Z$ holds due to $A : W$, i.e. the subroute of $A : B$ between A and W. To see it, note that $A : W$ is the only route between A and W that is minimally superactive wrt X, YW and Z in G. Now, consider the following two scenarios. The first scenario is when W is outside every rope in $A : B$. In this case, every recursive call that $X \sim_G YW|Z$ performs is of the form $A' \sim_{G_{\setminus A'}} XYZW \setminus AW$ with $A' \in Pa(A : W)$. This recursive call holds because $A' \in Pa(A : B)$ and, thus, $X \sim_G Y|Z$ performs the recursive call $A' \sim_{G_{\setminus A'}} XYZ \setminus AB$ and B and W are not used in it. The second scenario is when W is in some rope in $A : B$. In this case, $A : B$ has a subroute of the form $C \rightarrow D \rightarrow \ldots \rightarrow W \rightarrow \ldots \rightarrow E \leftarrow \ldots \leftarrow W \leftarrow \ldots \leftarrow D \leftarrow F$ with $C \neq F$ but possibly $D = W$ and/or $W = E$. However, in this case, $X \sim_G Y|ZW$ holds due to the route $(A : B) \setminus (W \rightarrow \ldots \rightarrow E \leftarrow \ldots \leftarrow W)$, here denoted ρ. To see it, note that ρ is the only route between A and B that is minimally superactive wrt X, Y and ZW in G. Moreover, every recursive call that $X \sim_G Y|ZW$ per-

forms is of the form $A' \sim_{G_{\setminus A'}} XYZW \setminus AB$ with $A' \in Pa(\rho)$. This recursive call holds because $A' \in Pa(A : B)$ and, thus, $X \sim_G Y|Z$ performs the recursive call $A' \sim_{G_{\setminus A'}} XYZ \setminus AB$ and W is not used in it. Finally, note that if $X \sim_G Y|ZW$ holds, then $X \sim_G YW|Z$ holds by weak union.

• Contraction1 $X \sim_G YW|Z \wedge X \perp_G Y|ZW \Rightarrow X \sim_G W|Z$.

In the proof of this property, we make use of the fact that separation in DAGs is a WT graphoid (Pearl, 1988) and, thus, it satisfies the ten properties introduced at the beginning of Section 4. Let $X \sim_G YW|Z$ hold due to a route $A : B$ with $A \in X$ and $B \in YW$. Then, $A \not\perp_G B|XYZW \setminus AB$ and, thus, $X \not\perp_G B(Y \setminus B)|WZ \setminus B$ by weak union. This implies that $B \notin Y$ because, otherwise, it would contradict $X \perp_G Y|ZW$. Likewise, for all $A' \in Pa(A : B)$, $A \not\perp_G A'|XYZW \setminus AA'$ and, thus, $X \setminus A' \not\perp_G A'(Y \setminus A')|WZ \setminus A'$ by weak union. This implies that $A' \notin Y$ because, otherwise, it would contradict $X \perp_G Y|ZW$. Furthermore, note that every recursive call that $X \sim_G YW|Z$ performs is of the form $A' \sim_{G_{\setminus A'}} XYZW \setminus AB$ with $A' \in Pa(A : B)$ and $A' \notin XYZW$. Assume that this recursive call holds due to a route $A' : B'$ with $B' \in XYZW \setminus AB$. By reasoning as above, we can conclude that $A' \not\perp_G B'|XYZW \setminus ABB'$ and $A' \not\perp_G A''|XYZW \setminus ABA''$ with $A'' \in Pa(A' : B')$. Then,

1. $A \not\perp_G A'|XYZW \setminus AB'$ by $A \not\perp_G A'|XYZW \setminus AA'$, $A' \notin XYZW$, and B' is not involved
2. $A' \not\perp_G B'|XYZW \setminus AB'$ by $A' \not\perp_G B'|XYZW \setminus ABB'$ and B is not involved
3. $A \not\perp_G B'|XYZW \setminus AB'$ by weak transitivity on (1), (2), and $A \not\perp_G B'|(XYZW \setminus AB')A'$
4. $X \setminus B' \not\perp_G B'(Y \setminus B')|WZ \setminus B'$ by weak union on (3).

Note that (4) implies that $B' \notin Y$ because, otherwise, it would contradict $X \perp_G Y|ZW$. Moreover,

5. $A \not\perp_G A'|XYZW \setminus AA''$ by $A \not\perp_G A'|XYZW \setminus AA'$, $A' \notin XYZW$, and A'' is not involved
6. $A' \not\perp_G A''|XYZW \setminus AA''$ by $A' \not\perp_G A''|XYZW \setminus ABA''$ and B is not involved
7. $A \not\perp_G A''|XYZW \setminus AA''$ by weak transitivity on (5), (6), and $A \not\perp_G A''|(XYZW \setminus AA'')A'$
8. $X \setminus A'' \not\perp_G B'(Y \setminus A'')|WZ \setminus A''$ by weak union on (7).

Note that (8) implies that $A'' \notin Y$ because, otherwise, it would contradict $X \perp_G Y|ZW$. Therefore, we have proven that $X \sim_G W|Z$ holds if the recursive calls performed by $X \sim_G YW|Z$ do not perform other recursive calls because, in this case, none of the key nodes is in Y and, thus, Y can be dropped. When a recursive call performed by $X \sim_G YW|Z$ performs another recursive call and this possibly another and so on, one just needs to repeat the reasoning above for each of these recursive calls.

• Contraction2 $X \sim_G YW|Z \wedge X \perp_G W|Z \Rightarrow X \sim_G Y|ZW$.

Let $X \sim_G YW|Z$ hold due to the route $A : B$ with $A \in X$ and $B \in YW$. We prove that $X \sim_G Y|ZW$ holds due to $A : B$. Since $A : B$ is minimally superactive wrt X, YW and Z in G, no node in $XYW \setminus AB$ can be in $A : B$. Then, $B \in Y$ by $X \perp_G W|Z$ and, thus, $A : B$ is minimally superactive wrt X, Y and ZW in G. Moreover, $A : B$ is the only such route between A and B. To see it, assume to the contrary that there is a second such route between A and B. Note that this second route must have some collider node in $C \in W$ for $A : B$ to be the only route between A and B that is minimally superactive wrt X, YW and Z in G. Then, this second route must also have some collider node $D \in Y$ between A and C by $X \perp_G W|Z$. However, this is a contradiction. Finally, note that every recursive call that $X \sim_G Y|ZW$ performs holds because $X \sim_G YW|Z$ also performs that recursive call because, as shown, $B \in Y$.

• Intersection $X \sim_G YW|Z \wedge X \perp_G Y|ZW \Rightarrow X \sim_G W|ZY$.

Let $X \sim_G YW|Z$ hold due to the route $A : B$ with $A \in X$ and $B \in YW$. We prove that $X \sim_G W|ZY$ holds due to $A : B$. Since $A : B$ is minimally superactive wrt X, YW and Z in G, no node in $XYW \setminus AB$ can be in $A : B$. Then, $B \in W$ by $X \perp_G Y|ZW$ and, thus, $A : B$ is minimally superactive wrt X, W and ZY in G. Moreover, $A : B$ is the only such route between A and B. To see it, assume to the contrary that

there is a second such route between A and B. Note that this second route must have some collider node in $C \in Y$ for $A : B$ to be the only route between A and B that is minimally superactive wrt X, YW and Z in G. Then, this second route must also have some non-collider node $D \in W$ between A and C by $X \perp_G Y|ZW$. However, this is a contradiction. Finally, note that every recursive call that $X \sim_G W|ZY$ performs holds because $X \sim_G YW|Z$ also performs that recursive call because, as shown, $B \in W$.

• Intersectional weak transitivity1 $X \sim_G V|ZY \wedge V \sim_G Y|ZX \wedge X \perp_G Y|Z \Rightarrow X \sim_G Y|ZV$.

Let $X \sim_G V|ZY$ and $V \sim_G Y|ZX$ hold due to the routes $A : V$ with $A \in X$ and $V : B$ with $B \in Y$, respectively. We prove that $A \sim_G B|XYZV \setminus AB$ holds, which implies $X \sim_G Y|ZV$ by weak union and decomposition. Note first that $X \perp_G Y|Z$ implies that $A : V$ followed by $V : B$, here denoted $A : B$, is the only route between A and B that is minimally superactive wrt A, B and $XYZV \setminus AB$ in G. Note also that every recursive call that $A \sim_G B|XYZV \setminus AB$ performs is of the form $A' \sim_{G_{\setminus A'}} XYZV \setminus AB$ with $A' \in Pa(A : B)$. Note also that $A' \in Pa(A : V)$ or $A' \in Pa(V : B)$. In the former case, we know that $X \sim_G V|ZY$ performs the recursive call $A' \sim_{G_{\setminus A'}} XYZ \setminus A$ which does not use B or V and, thus, $A' \sim_{G_{\setminus A'}} XYZV \setminus AB$ holds. In the latter case, we know that $V \sim_G Y|ZX$ performs the recursive call $A' \sim_{G_{\setminus A'}} XYZ \setminus B$ which does not use A or V and, thus, $A' \sim_{G_{\setminus A'}} XYZV \setminus AB$ holds.

• Intersectional weak transitivity2 $X \sim_G V|ZY \wedge V \sim_G Y|ZX \wedge X \perp_G Y|ZV \Rightarrow X \sim_G Y|Z$.

Let $X \sim_G V|ZY$ and $V \sim_G Y|ZX$ hold due to the routes $A : V$ with $A \in X$ and $V : B$ with $B \in Y$, respectively. We prove that $A \sim_G B|XYZ \setminus AB$ holds, which implies $X \sim_G Y|Z$ by weak union and decomposition. Note first that $X \perp_G Y|ZV$ implies that $A : V$ followed by $V : B$, here denoted $A : B$, is the only route between A and B that is minimally superactive wrt A, B and $XYZ \setminus AB$ in G. Note also that every recursive call that $A \sim_G B|XYZ \setminus AB$ performs is of the form $A' \sim_{G_{\setminus A'}} XYZ \setminus AB$ with $A' \in Pa(A : B)$. Note also that $A' \in Pa(A : V)$ or $A' \in Pa(V : B)$. In the former case, we know that $X \sim_G V|ZY$ performs the recursive call $A' \sim_{G_{\setminus A'}} XYZ \setminus A$ which does not use B and, thus, $A' \sim_{G_{\setminus A'}} XYZ \setminus AB$ holds. In the latter case, we know that $V \sim_G Y|ZX$ performs the recursive call $A' \sim_{G_{\setminus A'}} XYZ \setminus B$ which does not use A and, thus, $A' \sim_{G_{\setminus A'}} XYZ \setminus AB$ holds.

• Weak transitivity1 $X \sim_G V|Z \wedge V \sim_G Y|Z \wedge X \perp_G Y|Z \Rightarrow X \sim_G Y|ZV$.

1. $X \sim_G VY|Z$ and $VX \sim_G Y|Z$ by decomposition on $X \sim_G V|Z$ and $V \sim_G Y|Z$

2. $X \sim_G V|ZY$ and $V \sim_G Y|ZX$ by contraction2 on (1) and $X \perp_G Y|Z$

3. $X \sim_G Y|ZV$ by intersectional weak transitivity1 on (1), (2), and $X \perp_G Y|Z$.

• Weak transitivity2 $X \sim_G V|Z \wedge V \sim_G Y|Z \wedge X \perp_G Y|ZV \Rightarrow X \sim_G Y|Z$.

Just replace (3) in the proof of weak transitivity1 by

3. $X \sim_G Y|Z$ by intersectional weak transitivity2 on (1), (2), and $X \perp_G Y|ZV$. □

5 Discussion

As discussed in Section 1, the new criterion introduced in this paper is more (resp. less) powerful but less (resp. more) applicable than the criterion in (Bouckaert, 1995) (resp. (Peña, 2007)). To see it, consider Example 1. The new criterion and the criterion in (Bouckaert, 1995) can be applied but the criterion in (Peña, 2007) cannot, because p does not satisfy composition $X \perp_p Y|Z \wedge X \perp_p W|Z \Rightarrow X \perp_p YW|Z$. However, the new criterion reads $A \not\perp_p B|C$ from G but the criterion in (Bouckaert, 1995) does not, because A and B are not adjacent in G and this is necessary for that criterion to be conclusive.

References

Bouckaert, R. R. *Bayesian Belief Networks: From Construction to Inference*. PhD Thesis, University of Utrecht, 1995.

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

Peña, J. M. Reading Dependencies from Polytree-Like Bayesian Networks. In *UAI 2007*, 303-309.

Peña, J. M., Nilsson, R., Björkegren, J. and Tegnér, J. An Algorithm for Reading Dependencies from the Minimal Undirected Independence Map of a Graphoid that Satisfies Weak Transitivity. Journal of Machine Learning Research, 10, 1071-1094.

Studený, M. Bayesian Networks from the Point of View of Chain Graphs. In *UAI 1998*, 496-503.

Bayesian Network Sensitivity to Arc-Removal

Silja Renooij
Department of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
silja@cs.uu.nl

Abstract

Arc-removal is usually employed as part of an approximate inference scheme for Bayesian networks, whenever exact inference is intractable. We consider the removal of arcs in a different setting, as a means of simplifying a network under construction. We show how sensitivity functions, capturing the effects of parameter variation on an output of interest, can be employed to describe detailed effects of removing an arc. In addition, we provide new insights related to the choice of parameter settings upon arc removal, and the effect of this choice on the quality of the simplified model as an approximation of the original one.

1 Introduction

Arc removal is a model simplification technique most often employed as part of an approximate inference scheme for Bayesian networks. Whenever exact inference is intractable, a set of "weak" links is selected and removed to arrive at an approximate network in which exact inference is feasible (Kjærulff, 1994; Engelen, 1997; Choi, Chan & Darwiche, 2005).

In this paper, we consider the removal of arcs in a different setting, where we are interested in simplifying a model that is being constructed with the help of domain experts. A sparser model has the computational benefits of having fewer arcs and, hence, a smaller number of probability parameters. Furthermore, a sparser model may be easier to understand for the domain experts. Our focus is now on gaining detailed insight into the possible impact of removing a single, pre-selected arc on the behaviour of the network, both with and without evidence.

We will demonstrate that by interpreting arc removal as a constrained form of varying multiple parameters in the original network, we can study the effects of such a removal by means of so-called *sensitivity functions*. More specifically, a sensitivity-to-arc-removal function will describe the effects of *any* choice of setting the new parameters after arc removal, on some output probability of interest. For establishing these functions, we assume that inference in the original network is possible.

The quality of the new network can be assessed by evaluating its behaviour. Alternatively, when the quality of the new network as an *approximation* of the original one is of concern, the sensitivity-to-arc-removal functions can be plugged into some quality measure. In the context of arc removal, the quality of the approximation is usually measured in terms of the KL-divergence between the prior joint distributions of the original network and the approximate network (Kjærulff, 1994; Engelen, 1997). The KL-divergence has the convenient property that the change in prior joint distribution occasioned by the removal of an arc $A \rightarrow B$ can be computed locally from the probabilities for variable B and its parents (Kjærulff, 1994). This property does not necessarily hold, however, if we consider the KL-divergence between marginal distributions, or between posterior distributions. We will show that in this setting, such KL-divergences can be computed with our sensitivity functions.

The paper is structured as follows. Section 2 briefly reviews Bayesian networks and sensitivity functions. In Section 3 we derive the sensitivity-to-arc-removal functions. Section 4 demonstrates the use of these functions for computing KL-divergence; in doing so, some novel insights into the effect on KL-divergence of different choices of new parameters are given. The paper ends with conclusions and directions for future research in Section 5.

Pp. 233–241 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

2 Preliminaries

A Bayesian network compactly represents a joint probability distribution $\Pr$ over a set of stochastic variables $\mathbf{W}$ (Jensen & Nielsen, 2007). It combines an acyclic directed graph G, that captures the variables and their dependencies as nodes and arcs respectively, with conditional probability distributions $\Theta_{W_i|\pi(W_i)}$ for each variable W_i and its parents $\pi(W_i)$ in the graph, such that $\Pr(\mathbf{W}) = \prod_i \Theta_{W_i|\pi(W_i)}$.

We will refer to $\Theta_{W_i|\pi(W_i)}$ as the conditional probability table (CPT) of W_i; entries θ of Θ are called parameter probabilities, or parameters for short. In the remainder of this paper we will assume all variables to be binary-valued. Variables are denoted by capital letters and their values or instantiations by lower case; bold face is used for sets.

Probabilities computed from a Bayesian network are affected by the inaccuracies in the network's parameters. To investigate the extent of these effects, a sensitivity analysis can be performed in which $n \geq 1$ network parameters are varied simultaneously and the effect on an output probability of interest is studied. The effects of so-called n-*way* parameter variation are described by a *sensitivity function*. This is a multilinear function in the varied parameters in case of a prior probability of interest, and a rational function in the posterior case (Coupé & Van der Gaag, 2002). For example, the 2-*way* sensitivity function $f_{\Pr(a|\mathbf{e})}(x, y)$ describing the posterior probability $\Pr(a \mid \mathbf{e})$ as a function of two parameters x and y is given by

$$\frac{f_{\Pr(a\,\mathbf{e})}(x,y)}{f_{\Pr(\mathbf{e})}(x,y)} = \frac{c^{11}xy + c^{01}x + c^{10}y + c^{00}}{d^{11}xy + d^{01}x + d^{10}y + d^{00}}$$

where the constants c^{ij}, d^{ij}, $i, j \in \{0, 1\}$, are built from the non-varied parameters[1] in the network under study; feasible algorithms are available for their computation (Kjærulff & Van der Gaag, 2000; Coupé & Van der Gaag, 2002). Parameters from the same CPT $\Theta_{W_i|\pi(W_i)}$, but for different conditioning contexts, are independent; this results in zero interaction terms (Chan & Darwiche, 2004). In the above example this entails that $c^{11} = d^{11} = 0$.

[1] When a parameter θ varies as x, its complement $\overline{\theta} = 1 - \theta$ from the same distribution varies as $1 - x$. If θ concerns an k-valued variable, $k > 2$, then the $k - 1$ complementing parameters are co-varied proportionally.

3 Sensitivity Functions for Arc Removal

Sensitivity analysis typically refers to the study of effects of changes in network parameters on some outcome of interest. We can, however, also exploit it to study the effects of structural changes to the network's digraph, such as the removal of arcs.

Arc removal is most often employed as part of an approximate inference scheme, where arcs are removed until an approximate network is obtained in which exact inference is feasible (Kjærulff, 1994; Engelen, 1997; Choi, Chan & Darwiche, 2005). In this paper, we consider the removal of arcs in a different setting. We assume that we are constructing a Bayesian network with the help of domain experts, who are known to have the tendency of adding too many arcs into the model (Van der Gaag & Helsper, 2002). Our focus now is on studying the effects of removing a single arc, which we suspect may be superfluous, for the purpose of arriving at a simpler model that still suffices for the domain of application. We assume that inference in the original network is possible and, since we are still in a construction phase, that we have ample time to spend on it.

In this section, we propose the first approach for exactly studying the possible effects of arc removal on a probability of interest; the approach exploits the sensitivity function describing this probability in relation to the new parameters.

3.1 Implementing Arc Removal

Throughout this paper we consider the removal of an arc $A \rightarrow B$ from a Bayesian network $\mathcal{B}$, where $\pi(B) = \{A\} \cup \mathbf{Z}$. Removing an arc can be implemented in various ways (Choi, Chan & Darwiche, 2005). The approach we adopt in this paper is to simulate the removal by changes in the CPT $\Theta_{B|A\mathbf{Z}}$. More specifically, for each combination of values b and $\mathbf{z}$ the parameters $\theta_{b|a\mathbf{z}}$ are set to be equal for all values a; we will refer to these new parameters as $\theta'_{b|.\mathbf{z}}$, since the value of A is irrelevant.

The more or less standard approach to setting the new parameter values is by marginalising out variable A, or by approximating this process if it is infeasible to perform the exact computations (Engelen, 1997; Choi, Chan & Darwiche, 2005). More recent approaches focus on the addition of auxiliary nodes and parameters that compensate for the

lost dependency, together with iterative or variational methods that optimise these parameters as part of the arc-removal procedure (Choi & Darwiche, 2010; Choi & Darwiche, 2006). Rather than choosing a new parameter setting in advance, however, we can study the effects of all possible settings.

3.2 Sensitivity to Arc Removal

We study the effects of arc removal using a sensitivity analysis in which we vary all parameters $\theta_{b|a\mathbf{z}}$ until, for each b and context $\mathbf{z}$, the parameters are equivalent for all a. Let m be the number of different instantiations $\mathbf{z}$ for $\mathbf{Z}$, then arc removal requires the simultaneous variation of $2m$ parameters[2]. Generally, determining a $2m$-way sensitivity function is computationally demanding. The following proposition shows, however, that in the context of arc removal an m-way function suffices. Moreover, this function can be obtained from $2m$ 1-way sensitivity functions, which *can* be established efficiently (Kjærulff & Van der Gaag, 2000).

Proposition 1. *Let i index the values of variable A and j the instantiations of $\mathbf{Z}$. Let $x_{ij} = \theta_{b_1|a_i\mathbf{z}_j}$ and $1 - x_{ij} = \theta_{b_2|a_i\mathbf{z}_j}$ denote the $2 \cdot 2m$ parameters in $\Theta_{B|A\mathbf{Z}}$, and let $\Theta'_{B|.\mathbf{Z}}$ be the result of setting $x_{1j} = x_{2j}$ for all j. Let x'_j and $1-x'_j$ denote the parameters in Θ'. Then, the m-way sensitivity function which captures the effects of any possible choice for the parameters in $\Theta'_{B|.\mathbf{Z}}$ on an output probability of interest $\Pr(\mathbf{v})$ equals:*

$$f_{\Pr(\mathbf{v})}(x'_1, \ldots, x'_m) = \Big(\sum_{j=1}^{m}\Big(\sum_{i=1}^{2} c^1_{ij}\Big)\cdot x'_j\Big) + \Big(\sum_{i=1}^{2}\sum_{j=1}^{m} c^0_{ij}\Big) - (2\cdot m - 1)\cdot \Pr(\mathbf{v})$$

where c^0_{ij} and c^1_{ij} equal the constants from the 1-way sensitivity function describing $\Pr(\mathbf{v})$ as a function of parameter x_{ij}, $f_{\Pr(\mathbf{v})}(x_{ij}) = c^1_{ij}\cdot x_{ij} + c^0_{ij}$.

Proof: First we will detail the probabilistic semantics of the constants of a sensitivity function for $\Pr(\mathbf{v})$. Each of the $2m$ terms in the summation

$$\Pr(\mathbf{v}) = \sum_{i=1}^{2}\sum_{j=1}^{m} \Pr(\mathbf{v}\, a_i\mathbf{z}_j)$$

depends only on parameters $\theta_{B|a_i\mathbf{z}_j}$ with a corresponding conditioning context, and is constant with respect to other parameters in the CPT of B. More specifically, each $\Pr(\mathbf{v}\, a_i\mathbf{z}_j)$ relates to parameter $\theta_{b_1|a_i\mathbf{z}_j}$ as follows:

$$\begin{aligned}\Pr(\mathbf{v}\, a_i\mathbf{z}_j) &= \\ &= \sum_{k=1}^{2} \Pr(\mathbf{v} \mid b_k a_i\mathbf{z}_j)\cdot\Pr(b_k \mid a_i\mathbf{z}_j)\cdot\Pr(a_i\mathbf{z}_j) \\ &= \Pr(\mathbf{v} \mid b_1 a_i\mathbf{z}_j)\cdot\theta_{b_1|a_i\mathbf{z}_j}\cdot\Pr(a_i\mathbf{z}_j) \\ &\quad + \Pr(\mathbf{v} \mid b_2 a_i\mathbf{z}_j)\cdot(1-\theta_{b_1|a_i\mathbf{z}_j})\cdot\Pr(a_i\mathbf{z}_j)\end{aligned}$$

$\Pr(\mathbf{v})$ in terms of a single $\theta_{b_1|a_i\mathbf{z}_j}$ thus equals

$$\Pr(\mathbf{v}) = \big(c^1_{ij}\cdot\theta_{b_1|a_i\mathbf{z}_j} + r_{ij}\big) + \big(\Pr(\mathbf{v}) - \Pr(\mathbf{v}\, a_i\mathbf{z}_j)\big)$$

where

$$c^1_{ij} = \Big(\Pr(\mathbf{v} \mid b_1 a_i\mathbf{z}_j) - \Pr(\mathbf{v} \mid b_2 a_i\mathbf{z}_j)\Big)\cdot\Pr(a_i\mathbf{z}_j)$$

and $r_{ij} = \Pr(\mathbf{v} \mid b_2 a_i\mathbf{z}_j)\cdot\Pr(a_i\mathbf{z}_j)$. For $\Pr(\mathbf{v})$ in relation to $x_{ij} = \theta_{b_1|a_i\mathbf{z}_j}$ we therefore have a 1-way sensitivity function of the form $f_{\Pr(\mathbf{v})}(x_{ij}) = c^1_{ij}\cdot x_{ij} + c^0_{ij}$ with

$$c^0_{ij} = r_{ij} + \Pr(\mathbf{v}) - \Pr(\mathbf{v} a_i\mathbf{z}_j)$$

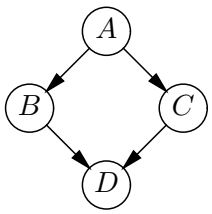
Consequently, upon varying all $2m$ parameters $x_{1j} = \theta_{b_1|a_1\mathbf{z}_j}$ and $x_{2j} = \theta_{b_1|a_2\mathbf{z}_j}$, $j = 1, \ldots, m$, we find from

$$\begin{aligned}\Pr(\mathbf{v}) &= \sum_{i=1}^{2}\sum_{j=1}^{m}\Big(c^1_{ij}\cdot\theta_{b_1|a_i\mathbf{z}_j} + r_{ij}\Big) \\ &= \sum_{i=1}^{2}\sum_{j=1}^{m}\Big(c^1_{ij}\cdot\theta_{b_1|a_i\mathbf{z}_j} + c^0_{ij}\Big) \\ &\quad - \sum_{i=1}^{2}\sum_{j=1}^{m}\Big(\Pr(\mathbf{v}) - \Pr(\mathbf{v}\, a_i\mathbf{z}_j)\Big)\end{aligned}$$

the function, $f_{\Pr(\mathbf{v})}(x_{11}, x_{21}, \ldots, x_{1m}, x_{2m}) =$

$$\begin{aligned}&= \Big(\sum_{i=1}^{2}\sum_{j=1}^{m} f_{\Pr(\mathbf{v})}(x_{ij})\Big) - (2\cdot m - 1)\cdot\Pr(\mathbf{v}) \\ &= \sum_{i=1}^{2}\sum_{j=1}^{m} c^1_{ij}\cdot x_{ij} + c^0\end{aligned}$$

[2] Another $2m$ parameters, for the other value of B, are covaried.



$\theta_{a_1} = 0.8$

$\theta_{b_1|a_1} = 0.8$
$\theta_{b_1|a_2} = 0.4$

$\theta_{c_1|a_1} = 0.5$
$\theta_{c_1|a_2} = 1.0$

$\theta_{d_1|b_1c_1} = 0.1$
$\theta_{d_1|b_1c_2} = 0.3$
$\theta_{d_1|b_2c_1} = 0.9$
$\theta_{d_1|b_2c_2} = 0.8$

Figure 1: The example Bayesian network, taken from Choi & Darwiche (2010).

with $c^0 = \Big(\sum_{i=1}^{2}\sum_{j=1}^{m} c^0_{ij}\Big) - (2{\cdot}m - 1){\cdot}\Pr(\mathbf{v})$

The above $2m$-way sensitivity function describes all possible effects of varying the $2m$ parameters on $\Pr(\mathbf{v})$. To study the effects of arc removal, however, variation of these parameters is constrained in the sense that $x_{1j} = x_{2j}$ should hold for each $j = 1, \ldots, m$. That is, rather than a $2m$ dimensional parameter space, we are actually dealing with an m dimensional space and an m-way sensitivity function. Using x'_j to denote the parameters $\theta'_{b_1|.\mathbf{z}_j}$ resulting from setting $x_{1j} = x_{2j}$, we conclude

$$f_{\Pr(\mathbf{v})}(x'_1, \ldots, x'_m) = \Big(\sum_{j=1}^{m}\Big(\sum_{i=1}^{2} c^1_{ij}\Big){\cdot}x'_j\Big) + c^0$$

□

Note that in the above proposition we have not assumed $\Pr(\mathbf{v})$ to be a marginal over a single variable. The proposition is therefore more generally applicable, but most algorithms for computing the constants of sensitivity functions assume that we are interested in a single-variable marginal (prior or posterior). Generalisation of the proposition to a posterior probability of interest, such as $\Pr(\mathbf{v} \mid \mathbf{e}) = \frac{\Pr(\mathbf{ve})}{\Pr(\mathbf{e})}$, is also straightforward, as we demonstrate in the following example.

Example 1. Consider the Bayesian network in Figure 1, which will serve as a running example throughout the paper. We assume that variable A can take on two values, represented by a_1 and a_2, respectively. A similar assumption holds for the other variables in the network. We are interested in studying the effects of removing arc $A \rightarrow B$, and therefore consider the parameters $x_1 = \theta_{b_1|a_1}$ and $x_2 = \theta_{b_1|a_2}$, and their complements.

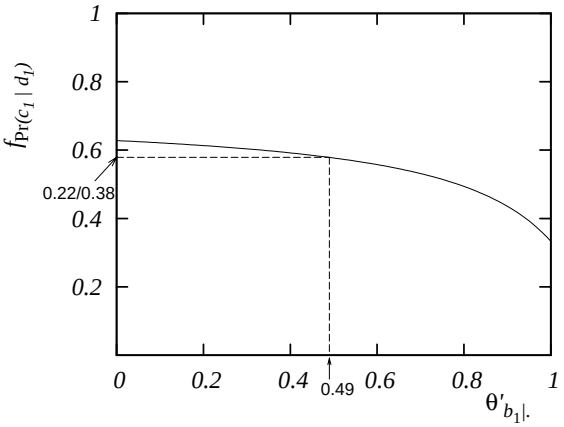


Figure 2: $\Pr(c_1 \mid d_1)$ as a function of the new parameters $\theta'_{b_1|.}$ and $\theta'_{b_2|.} = 1 - \theta'_{b_1|.}$.

Suppose $\Pr(c_1 \mid d_1)$ is our probability of interest, with original value $\Pr(c_1 d_1)/\Pr(d_1) = 0.22/0.38$. The relevant 1-way sensitivity functions are:

$$f_{\Pr(c_1|d_1)}(x_1) = \frac{-0.32{\cdot}x_1 + 0.48}{-0.52{\cdot}x_1 + 0.80}$$

$$f_{\Pr(c_1|d_1)}(x_2) = \frac{-0.16{\cdot}x_2 + 0.28}{-0.16{\cdot}x_2 + 0.44}$$

Following Proposition 1, simultaneous variation of x_1 and x_2 results in: $f_{\Pr(c_1|d_1)}(x_1, x_2) =$

$$= \frac{-0.32{\cdot}x_1 - 0.16{\cdot}x_2 + 0.48 + 0.28 - 0.22}{-0.52{\cdot}x_1 - 0.16{\cdot}x_2 + 0.80 + 0.44 - 0.38}$$

$$\Longrightarrow \frac{-0.48{\cdot}x' + 0.54}{-0.68{\cdot}x' + 0.86} = f_{\Pr(c_1|d_1)}(x')$$

where x' results from setting $x_1 = x_2$. This function, shown in Figure 2, now describes the possible effects of removing arc $A \rightarrow B$ on the probability $\Pr(c_1 \mid d_1)$. From the function we can, for example, compute which new parameter setting will result in the original value of the probability of interest:

$$f_{\Pr(c_1|d_1)}(x') = \frac{0.22}{0.38} \iff x' = 0.49 \qquad \square$$

Although we assumed all variables to be binary, Proposition 1 trivially extends to non-binary variables A and $\mathbf{Z}$. If variable B can take on $n > 2$ values, however, the sensitivity function describing the effects of arc removal can no longer be obtained from 1-way sensitivity functions. In that case, varying the parameters $\theta_{b_1|a_i\mathbf{z}}$ until they become equal for all a_i, no longer ensures that all proportionally

co-varying parameters $\theta_{b_2|a_i\mathbf{z}} \ldots \theta_{b_n|a_i\mathbf{z}}$, $n > 2$, become equal for all a_i. To enforce such equalities, $(n-1)$-way analyses are necessary.

By studying the sensitivity functions that describe the effects of arc removal on an output of interest for *various outputs* and for *various combinations of observations*, we can determine whether there exists a parameter setting that results in acceptable behaviour of the simplified model. From the sensitivity functions we can, for example, immediately determine if a specific case entered into the network would result in the same most likely value of our output variable in both the original and the simplified network. Another way to compare the models, is by comparing the distributions they define.

4 Arc Removal and KL-divergence

Empirical evidence shows that arc removal can lead to quite a speedup in inference, at the cost of only little deterioration in quality (Choi, Chan & Darwiche, 2005; Santana & Provan, 2008; Choi & Darwiche, 2010). In this context, quality is measured in terms of the Kullback-Leibler (KL) divergence between the original distribution $\Pr$ and the distribution $\Pr'$ for the approximate network (Cover & Thomas, 1991); it is defined by

$$\mathrm{KL}(\Pr(\mathbf{V}), \Pr'(\mathbf{V})) \stackrel{\mathbf{def}}{=} \sum_{\mathbf{v}} \Pr(\mathbf{v}) \cdot \log \frac{\Pr(\mathbf{v})}{\Pr'(\mathbf{v})}$$

The KL-divergence has the convenient property that the change in *prior joint* distribution occasioned by the removal of an arc $A \rightarrow B$ can be computed locally from the probabilities for variable B and its parents (Kjærulff, 1994). This property does not necessarily hold, however, if we consider the KL-divergence between marginal distributions, or between posterior distributions, which are typically of interest for practical applications. Recently, it was argued that arc-removal methods should take available evidence into account, in order to tailor the approximation to the evidence at hand (Choi, Chan & Darwiche, 2005; Choi & Darwiche, 2010). In this section we will consider KL-divergence as a function of the new parameter settings, and demonstrate that we can plug in our sensitivity-to-arc-removal functions in order to compute this divergence, both between joint and marginal distributions, with and without evidence.

4.1 Joint Prior and Joint Posterior Divergence

If, upon arc removal, we wish to choose the new parameter settings such that they minimise the KL-divergence between the original and the simplified network, then under some conditions this choice is evident. The clear-cut cases concern joint (prior or posterior) distributions and are given by the proposition below.

In the remainder of this section we let network $\mathcal{B}'$ be the result of removing arc $A \rightarrow B$ from the original network $\mathcal{B}$. The distribution defined by $\mathcal{B}'$ is denoted $\Pr'$.

Proposition 2. *Consider the two joint prior distributions* $\Pr(\mathbf{V})$ *and* $\Pr'(\mathbf{V})$*, and two joint posterior distributions* $\Pr(\mathbf{V} \mid \mathbf{e})$ *and* $\Pr'(\mathbf{V} \mid \mathbf{e})$ *conditioned on evidence* $\mathbf{e}$*. Then*

- $\mathrm{KL}(\Pr(\mathbf{V}), \Pr'(\mathbf{V}))$ *is minimised by setting, for all* b *and* $\mathbf{z}$ *combinations,* $\theta'_{b|.\mathbf{z}} = \Pr(b \mid \mathbf{z})$*;*
- $\mathrm{KL}(\Pr(\mathbf{V} \mid \mathbf{e}), \Pr'(\mathbf{V} \mid \mathbf{e}))$ *is minimised by setting, for all* b *and* $\mathbf{z}$ *combinations,* $\theta'_{b|.\mathbf{z}} = \Pr(b \mid \mathbf{z}\mathbf{e})$*, if* $\Pr(\mathbf{e}) = \Pr'(\mathbf{e})$.

Proof: The factorisation of the joint distribution is exploited to reduce the KL-divergence to terms involving the CPT of variable B (see Kjærulff (1994) or Engelen (1997) for the prior situation and Choi, Chan & Darwiche (2005) for the posterior case):

$$\begin{aligned}
&\mathrm{KL}(\Pr(\mathbf{V} \mid \mathbf{e}), \Pr'(\mathbf{V} \mid \mathbf{e})) = \\
&\quad = \sum_{\mathbf{v}} \Pr(\mathbf{v} \mid \mathbf{e}) \cdot \log \frac{\Pr(\mathbf{v} \mid \mathbf{e})}{\Pr'(\mathbf{v} \mid \mathbf{e})} \\
&\quad = \log \frac{\Pr'(\mathbf{e})}{\Pr(\mathbf{e})} + \sum_{ab\mathbf{z}} \Pr(ab\mathbf{z} \mid \mathbf{e}) \cdot \log \frac{\theta_{b|a\mathbf{z}}}{\theta'_{b|.\mathbf{z}}}
\end{aligned}$$

The term $\sum_{ab\mathbf{z}} \Pr(ab\mathbf{z} \mid \mathbf{e}) \cdot \log \theta_{b|a\mathbf{z}}$ is determined by the original network only. The remaining terms are a function of the new parameters and equals:

$$\begin{aligned}
&\log \frac{\Pr'(\mathbf{e})}{\Pr(\mathbf{e})} - \sum_{ab\mathbf{z}} \Pr(ab\mathbf{z} \mid \mathbf{e}) \cdot \log \theta'_{b|.\mathbf{z}} = \\
&= \log \frac{\Pr'(\mathbf{e})}{\Pr(\mathbf{e})} + \sum_{\mathbf{z}} \Pr(\mathbf{z} \mid \mathbf{e}) \cdot \\
&\qquad \cdot \Big(- \sum_{b} \Pr(b \mid \mathbf{z}\mathbf{e}) \cdot \log \theta'_{b|.\mathbf{z}} \Big)
\end{aligned}$$

The bracketed summation equals the cross-entropy between the two distributions over B and is known to be minimal if the distributions are the same, i.e. $\theta'_{b|.\mathbf{z}} = \Pr(b \mid \mathbf{ze})$.

In the prior situation, we get the same formula but without the $\log(\Pr'(\mathbf{e})/\Pr(\mathbf{e}))$ term, and with the $\mathbf{e}$'s removed from the conditioning contexts. In that case, minimising cross-entropy, i.e. setting $\theta'_{b|.\mathbf{z}} = \Pr(b \mid \mathbf{z})$, serves to minimise the KL-divergence. In the posterior case, however, minimising cross-entropy is only *guaranteed* to minimise the KL-divergence if $\Pr'(\mathbf{e}) = \Pr(\mathbf{e})$, i.e. if the probability of evidence is insensitive to changes in the parameters for B. □

The first property in the above proposition, although to the best of our knowledge never explicitly proven, must be well-known: the optimal parameter setting stated amounts exactly to marginalising out variable A, which is a standard approach to implementing arc removal.

For the two cases stated in Proposition 2, we have an expression defining the KL-divergence in relation to the new parameter settings. In case $\Pr(\mathbf{e}) \neq \Pr'(\mathbf{e})$, we can now plug in the sensitivity function $f_{\Pr'(\mathbf{e})}(\theta'_{b|.\mathbf{z}})$ and again get the KL-divergence as a function of the new parameters. For low-dimensional functions it is then easy to compute the parameter settings that minimise the divergence.

Example 2. Reconsider the example Bayesian network in Figure 1. We will use the terms *prior KL-divergence* and *posterior KL-divergence* to refer to the divergence between (joint) prior and posterior distributions, respectively. The prior KL-divergence as a function of $x' = \theta'_{b_1|.}$ equals

$$\begin{aligned}\mathrm{KL}(\Pr(ABCD), \Pr'(ABCD))(x') = \\ = \sum_{i=1}^{2}\sum_{j=1}^{2} \Pr(a_i b_j)\cdot\log\theta_{b_j|a_i} + \\ - \Pr(b_1)\cdot\log x' - \Pr(b_2)\cdot\log(1-x') \\ = -0.77 - 0.72\cdot\log x' - 0.28\cdot\log(1-x')\end{aligned}$$

and is shown in Figure 3 (dashed). We can indeed verify from the figure that the values of the new parameters that correspond with the marginal probabilities for B, i.e. $\theta'_{b_1|.} = 0.80\cdot0.8 + 0.4\cdot0.2 = 0.72$ and $\theta'_{b_2|.} = 0.28$, result in a minimal KL-divergence

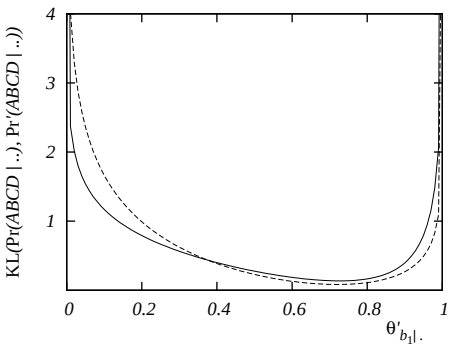


Figure 3: The prior KL-divergence (dashed), and the posterior divergence, given evidence d_1, as a function of the new parameters $\theta'_{b_1|.}$ and $\theta'_{b_2|.} = 1 - \theta'_{b_1|.}$ (zero and one excluded).

of 0.08. Figure 3 also shows the joint *posterior* KL-divergence in the context of evidence d_1 (solid); this can be written in terms of x' by using the sensitivity function $f_{\Pr'(d_1)}(x')$:

$$\begin{aligned}\mathrm{KL}(\Pr(ABC \mid d_1), \Pr'(ABC \mid d_1))(x') = \\ = \log\frac{-0.68\cdot x' + 0.86}{0.38} + 0.52 \\ - 0.28\cdot\log x' - 0.24\cdot\log(1-x')\end{aligned}$$

Taking the first derivative, we find that this function is minimised for $x' = 0.73$. This same value for $\Pr(b_1)$ follows from the auxiliary parameters established, for this *same* example network, by the iterative procedure in Choi & Darwiche (2010). Note that this optimal value is found for a much higher value of x' than would be found through marginalisation, i.e. $\theta'_{b|.\mathbf{z}} = \sum_a \Pr(b \mid a\mathbf{ze})\cdot\Pr(a \mid \mathbf{ze})$:

$$\theta'_{b_1|.} = 0.48\cdot0.69 + 0.07\cdot0.31 = 0.36$$

This is caused by the fact that $\Pr(d_1)$ is sensitive to the parameter changes. The parameter setting used in (Choi, Chan & Darwiche, 2005), $\theta'_{b|.\mathbf{z}} = \sum_a \theta_{b|a\mathbf{z}}\cdot\Pr(a \mid \mathbf{e})$, results in:

$$\theta'_{b_1|.} = 0.8\cdot0.69 + 0.4\cdot0.31 = 0.68$$

Remarkably, these settings are closer to the optimum, despite their use of the invalid independence assumption that B is independent of D given A. □

4.2 Marginal Prior and Posterior Divergence

Suppose we wish to choose the new parameter settings such that they minimise the KL-divergence between marginal rather than joint distributions. The

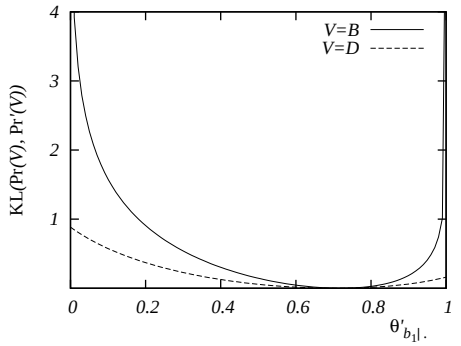


Figure 4: Prior marginal KL-divergences, per variable, as a function of the new parameters $\theta'_{b_1|.}$ and $\theta'_{b_2|.} = 1 - \theta'_{b_1|.}$. For variables A and C the divergence is zero.

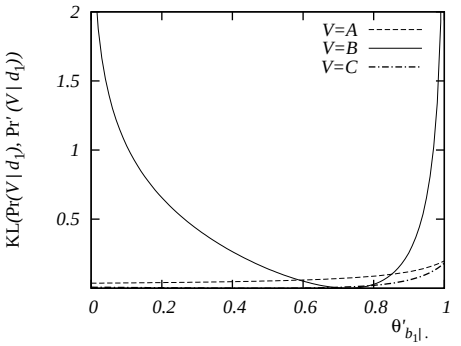


Figure 5: Posterior marginal KL-divergences per variable, given evidence d_1, as a function of the new parameters $\theta'_{b_1|.}$ and $\theta'_{b_2|.} = 1 - \theta'_{b_1|.}$.

KL-divergence between marginal distributions is hardly ever considered, since in that case we cannot exploit the factorisation of the joint distribution to reduce the divergence to local terms. Using the sensitivity functions for arc removal introduced in the previous section, however, we can define the KL-divergence as a function of the new parameters under consideration.

Corollary 1. *Let* $\Pr$, $\Pr'$ *and* $x'_1, \ldots x'_m$ *be as before, then*

$$\mathrm{KL}(\Pr(\mathbf{V} \mid \mathbf{e}), \Pr'(\mathbf{V} \mid \mathbf{e}))(x'_1, \ldots x'_m) =$$

$$= \sum_{\mathbf{v}} \Pr(\mathbf{v} \mid \mathbf{e}) \cdot \log \frac{\Pr(\mathbf{v} \mid \mathbf{e})}{f_{\Pr'(\mathbf{v}|\mathbf{e})}(x'_1, \ldots x'_m)}$$

Note that the above corollary applies to both joint and marginal distributions; in the prior situation, the above holds with all occurrences of $\mathbf{e}$ removed. Its formula in fact was used to create the graphs of Figures 3, 4 and 5.

The following example illustrates, for each variable V in our example network, the KL-divergence between the original marginal distribution $\Pr(V)$ and the new marginal distribution $\Pr'(V)$, for different choices of the new parameters for variable B. We will refer to these divergences as *marginal KL-divergences*.

Example 3. Reconsider the example Bayesian network in Figure 1, from which we remove arc $A \rightarrow B$. Figure 4 now shows the prior marginal KL-divergences for each variable and all possible choices for the new parameters θ'_{b_i} (zero and one excluded). Since the values in the CPT for variable B affect only the marginal distributions of B and D, the KL-divergences for A and C are zero. In addition, we see that the parameter settings that optimise the marginal KL-divergence for B, also optimise the divergence for its descendant D.

Figure 5 similarly shows the various marginal KL-divergences in the context of evidence d_1. We see that the parameter setting for which the KL-divergence is optimal, now varies per variable: this is due to the fact that the marginal KL-divergence for a certain variable is optimal, if the new parameters are chosen such that the new marginal probabilities equal the original ones.

As an illustration, suppose we are interested in variable C, in the context of evidence d_1. Recall that the sensitivity function for c_1 is given by

$$f_{\Pr'(c_1|d_1)}(x') = \frac{-0.48 \cdot x' + 0.54}{-0.68 \cdot x' + 0.86}$$

The original value for $\Pr(c_1 \mid d_1)$ equals $\frac{0.22}{0.38}$, which we can obtain in our new network by setting $x' = \theta'_{b_1|.} = 0.49$. Figure 5 indeed suggests that this choice is optimal in terms of KL-divergence. □

From the previous examples we have that the optimal choice for the new parameter settings, in terms of minimising the KL-divergence, differs between prior and posterior distributions, both for joint and marginal distributions. In the example network, however, setting the new parameter $\theta'_{b_1|.}$ to a value somewhere in the range $[0.6, 0.8]$ seems to result in a small KL-divergence in all situations considered.

5 Conclusion

In this paper we introduced sensitivity functions as a means of studying the exact impact of removing an arc from a Bayesian network on some output of interest. These functions provide insight into whether or not removing the arc can result in an acceptable simplified model, and they can support our choice for setting the new parameters upon removal. If the simplified network should be a good quality approximation of the original one, then the sensitivity functions can also be used to find new parameters that minimise the KL-divergence between various distributions of interest.

In addition, we provided some insights concerning arc removal and KL-divergence. More specifically, we showed that arc removal by means of marginalisation is in fact optimal in terms of minimising the KL-divergence between prior joint distributions. Secondly, we provided a condition under which marginalisation results in an optimal KL-divergence between posterior joint distributions.

We assumed that all variables are binary-valued. As mentioned, extension to non-binary variables is trivial, except for variable B. For non-binary B, proportional co-variation of its values no longer ensures that all parameters that should be equated for arc removal in fact are. As a result, multi-way functions with non-zero interaction terms are necessary. Further research is required to establish the exact implications of this increased complexity.

Although our interest in arc removal is not in approximating networks to make inference feasible, our results can be put to use in situations where the complexity of a network is such that exact inference is possible, but too time-consuming for practical purposes. In such a case, detailed insights concerning the effects of arc removal can be obtained prior to deploying the network, and then exploited to construct efficient approximations for certain sets of observable variables, or for varying output variables of interest.

Acknowledgement

I would like to thank Linda van der Gaag for our inspiring discussions and the anonymous reviewers for useful comments.

References

H. Chan, A. Darwiche (2004). Sensitivity analysis in Bayesian networks: from single to multiple parameters. *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, AUAI Press, pp. 67 – 75.

A. Choi, H. Chan, A. Darwiche (2005). On Bayesian network approximation by edge deletion. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, AUAI Press, pp. 128 – 135.

A. Choi, A. Darwiche (2006). A variational approach for approximating Bayesian networks by edge deletion. *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, AUAI Press, pp. 80 – 89.

A. Choi, A. Darwiche (2010). An edge deletion semantics for belief propagation. *Submitted for publication*. Available as `http://reasoning.cs.ucla.edu/fetch.php?id=98&type=pdf`

V.M.H. Coupé, L.C. van der Gaag (2002). Properties of sensitivity analysis of Bayesian belief networks. *Annals of Mathematics and Artificial Intelligence*, 36, pp. 323 – 356.

Th.M. Cover, J.A. Thomas (1991). *Elements of Information Theory*, Wiley-Interscience.

R.A. van Engelen (1997). Approximating Bayesian belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, pp. 916 – 920.

F.V. Jensen, T.D. Nielsen (2007). *Bayesian Networks and Decision Graphs*, Springer-Verlag.

U. Kjærulff (1994). Reduction of computational complexity in Bayesian networks through removal of weak dependencies. *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pp. 374 – 382.

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, pp. 317 – 325.

A. Santana, G. Provan (2008). An analysis of Bayesian network model-approximation techniques. *Proceedings of the 18th European Conference on Artificial Intelligence*, IOS Press, pp. 851 – 852.

L.C. van der Gaag, E.M. Helsper (2002). Experiences with modelling issues in building probabilistic networks. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, LNCS 2473, Springer-Verlag, pp. 21 – 26.

Efficient Sensitivity Analysis in Hidden Markov Models

Silja Renooij
Department of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
silja@cs.uu.nl

Abstract

Sensitivity analysis in a Hidden Markov model (HMM) usually amounts to applying a change to its parameters and re-computing its output of interest. Recently it was shown that, as in Bayesian networks, a simple mathematical function describes the relation between a model parameter and a probability of interest in an HMM. Up till now, however, no special purpose algorithms existed for determining this function. In this paper we present a new and efficient algorithm for doing so, which exploits the recursive properties of an HMM.

1 Introduction

Hidden Markov models (HMMs) are frequently applied statistical models for capturing processes that evolve over time. An HMM can be represented by the simplest type of Dynamic Bayesian network (see for details Smyth, Heckerman & Jordan, 1997; Murphy (2002)), which entails that all sorts of algorithms available for (Dynamic) Bayesian networks can be straightforwardly applied to HMMs.

HMMs specify a number of parameter probabilities, which are bound to be inaccurate to at least some degree. Sensitivity analysis is a standard technique for studying the effects of parameter inaccuracies on the output of a model. An analysis in which a single parameter is varied, is called a *one-way* sensitivity analysis; in an n*-way* analysis $n > 1$ parameters are varied simultaneously. For Bayesian networks, a simple mathematical function exists that describes the relation between one or more network parameters and an output probability of interest. Various algorithms are available for computing the constants of this so-called sensitivity function (see Coupé et al. (2000) for an overview and comparison of existing algorithms). Recently, it was shown that similar functions describe the relation between model parameters and output probabilities in HMMs (Charitos & Van der Gaag, 2004). For computing the constants of these functions, it was suggested to represent the HMM as a Bayesian network, unrolled for a fixed number of time slices, and to use the above mentioned algorithms for computing the constants of the sensitivity function. The drawback of this approach is that the repetitive character of the HMM, with the same parameters occurring for each time step, is not exploited in the computation of the constants. As such, using standard Bayesian network algorithms may not be the most efficient approach to determining sensitivity functions for HMMs.

In this paper we present a new and efficient algorithm for computing the constants of the sensitivity function in HMMs, which exploits the recursive properties of an HMM. After presenting some preliminaries concerning HMMs and sensitivity functions in Section 2, we review the known recursive expressions for different probabilities of interest in Sections 3 and 4; more specifically, we focus on so-called filter and prediction probabilities in Section 3 and on smoothing in Section 4. In these sections, we subsequently translate the recursive expressions into functions of model parameters and present algorithms for computing the constants of the associated sensitivity functions. We discuss relevant related work in Section 5 and conclude the paper with directions for future research in Section 6.

2 Preliminaries

For each time t, an HMM consists of a single hidden variable whose state can be observed by some test or sensor. The uncertainty in the test or sensor

Pp. 241–249 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

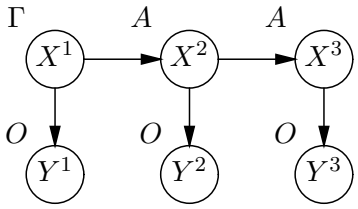


Figure 1: A Bayesian network representation of an HMM unrolled for three time slices.

output is captured by a set of observation probabilities; the transitions among the states in subsequent time steps, or *time slices*, are captured by a set of transition probabilities. In this paper we concentrate on HMMs with discrete observable variables. We further assume that the model is time-invariant, i.e. the probability parameters do not depend on time. More formally, an HMM now is a statistical model $H = (X, Y, A, O, \Gamma)$, where for each time $t \geq 1$:

- X^t is the hidden variable; its states are denoted by x_i^t, $i = 1, \ldots, n$, $n \geq 2$;
- Y^t is the observable variable, with states denoted by y_j^t, $j = 1, \ldots, m$, $m \geq 2$; the notation y_e^t is used to indicate actual evidence;
- A is the transition matrix with entries $a_{i,j} = p(x_j^{t+1} \mid x_i^t)$, $i, j = 1, \ldots, n$;
- O is the observation matrix with entries $o_{i,j} = p(y_j^t \mid x_i^t)$, $i = 1, \ldots, n$, $j = 1, \ldots, m$;
- Γ is the initial vector for X^1 with entries $\gamma_i = p(x_i^1)$, $i = 1, \ldots, n$.

Figure 1 shows a Bayesian network representation of an HMM unrolled for three time slices.

Inference in temporal models typically amounts to computing the marginal distribution over X at time t, given the evidence up to and including time T, that is $p(X^t \mid y_e^{1:T})$, where $y_e^{1:T}$ is short for the sequence of observations $y_e^1, \ldots, y_e^T$. If $T = t$, this inference task is known as *filtering*, $T < t$ concerns *prediction* of a future state, and *smoothing* is the task of inferring the past, that is $T > t$. For exact inference in an HMM, the efficient Forward-Backward algorithm is available (see for details Russel & Norvig (2003, chapter 15)). This algorithm computes for all hidden states i at time t, the following two probabilities:

- forward probability $F(i,t) = p(x_i^t, y_e^{1:t})$, and
- backward probability $B(i,t) = p(y_e^{t+1:T} \mid x_i^t)$

resulting in

$$p(x_i^t \mid y_e^{1:T}) = \frac{p(x_i^t, y_e^{1:T})}{p(y_e^{1:T})} = \frac{F(i,t) \cdot B(i,t)}{\sum_{i=1}^{n} F(i,t) \cdot B(i,t)}$$

Alternatively, the HMM can be represented as a Bayesian network unrolled for $\max\{t, T\}$ time slices, upon which standard Bayesian network inference algorithms can be used.

The outcome $p(x_i^t \mid y_e^{1:T})$ depends on the probability parameters specified for the model. To study the effects of possible inaccuracies in these parameters on the computed output, a *sensitivity analysis* can be done. To this end, we establish the *sensitivity function* $p(x_i^t \mid y_e^{1:T})(\theta)$ that describes our output of interest in terms of parameter θ, where θ can be any model parameter, i.e. an initial probability, an observation probability or a transition probability.[1]

In the context of Bayesian networks, sensitivity analysis has been studied extensively by various researchers (see Van der Gaag, Renooij & Coupé (2007) for an overview and references). In the context of HMMs, sensitivity analysis is usually performed by means of a perturbation analysis where a small change is applied to the parameters, upon which the output of interest is re-computed (Mitrophanov, Lomsadze & Borodovsky, 2005). The main difference between sensitivity analysis in Bayesian networks and in Hidden Markov models in essence is that a single parameter in an HMM may occur multiple times. A one-way sensitivity analysis in an HMM, therefore, amounts to an n-way analysis in its Bayesian network representation, where n equals the number of time slices under consideration. It is therefore no surprise that for HMMs sensitivity functions are similar to those for Bayesian networks (Charitos & Van der Gaag, 2004). The difference with the general n-way function for Bayesian networks is, however, that the n parameters are constrained to

[1] If a parameter $\theta = p(v_j \mid \pi)$ for a variable V is varied, we must ensure that still $\sum_i p(v_i \mid \pi) = 1$. To this end, all probabilities $p(v_i \mid \pi)$, $i \neq j$, are co-varied proportionally: $p(v_i \mid \pi)(\theta) = p(v_i \mid \pi) \cdot \frac{1-\theta}{1-p(v_j \mid \pi)}$. For binary-valued V this simplifies to $p(v_i \mid \pi)(\theta) = 1 - \theta$.

all be equal, which reduces the number of required constants. We now summarise the known results for sensitivity functions in HMMs (Charitos & Van der Gaag, 2004; Charitos, 2007). For the probability of evidence as a function of a model parameter θ, we have the following polynomial function:

$$p(y_e^{1:T})(\theta) = d_N^T \cdot \theta^N + \ldots + d_1^T \cdot \theta + d_0^T$$

where $N = T$ if $\theta = o_{r,s}$, $N = T-1$ if $\theta = a_{r,s}$, $N = 1$ for $\theta = \gamma_r$, and coefficients d_N^T, …, d_0^T are constants with respect to the various parameters. For the joint probability of a hidden state and evidence, as a function of a model parameter θ, we have the following polynomial function:

$$p(x_v^t,\, y_e^{1:T})(\theta) = c_{v,N}^t \cdot \theta^N + \ldots + c_{v,1}^t \cdot \theta + c_{v,0}^t$$

where

$$N = \begin{cases} t-1 & \text{if } \theta = a_{r,s} \text{ and } t \geq T; \\ T & \theta = o_{r,s} \text{ and } v = r; \\ T-1 & \theta = o_{r,s} \text{ and } v \neq r, \text{ or} \\ & \theta = a_{r,s}, t < T \text{ and } v = r; \\ T-2 & \theta = a_{r,s}, t < T \text{ and } v \neq r; \\ 1 & \theta = \gamma_r; \end{cases}$$

and coefficients $c_{v,N}^t$, …, $c_{v,0}^t$ are constants with respect to the various parameters. The same general forms apply to prior marginals over X, by taking $T = 0$. Note that prior probabilities are not affected by variation in observation parameters $o_{r,s}$.

Up till now, no special purpose algorithms for establishing the coefficients of the sensitivity functions in an HMM were available, which means that Bayesian network algorithms need to be used to this end. In the next sections, we present a new and efficient algorithm for computing the coefficients for sensitivity functions in HMMs. To this end, we exploit the repetitive character of the model parameters and knowledge about the polynomial form of the sensitivity functions presented above. We will discuss the inference tasks filtering/prediction and smoothing separately.

3 Filter and Prediction Coefficients

In this section we establish the coefficients of the sensitivity function $p(x_v^t \mid y_e^{1:T})(\theta)$, $t \geq T$, for various model parameters θ. Note that the sensitivity function for a probability $p(x_v^t \mid y_e^{1:T})$ is a quotient of the sensitivity functions for $p(x_v^t,\, y_e^{1:T})$ and for $p(y_e^{1:T})$, and that $p(y_e^{1:T}) = \sum_{z=1}^n p(x_z^t,\, y_e^{1:T})$. Therefore, given the polynomial form of the functions, the coefficients for the sensitivity functions for $p(x_v^t,\, y_e^{1:T})$ provide enough information to establish all required coefficients. The remainder of this section is therefore devoted to obtaining the coefficients for $p(x_v^t,\, y_e^{1:T})$ as a function of θ.

3.1 Filter and Prediction Recursions

We will now review the recursive expression for filter probabilities (see for details Russel & Norvig (2003, chapter 15)) and make explicit the relation between filter and prediction probabilities. Starting with the latter, we find by conditioning on X^T and exploiting the independence $X^t \perp Y^{1:T} \mid X^T$ for $T < t$, that

$$p(x_v^t,\, y_e^{1:T}) = \sum_{z=1}^n p(x_v^t \mid x_z^T) \cdot p(x_z^T,\, y_e^{1:T}) \quad (1)$$

The second factor in this summation is a filter probability; the first can be computed from similar probabilities in time slice $t-1$ for $t > T+1$:

$$p(x_v^t \mid x_z^T) \;=\; \sum_{w=1}^n a_{w,v} \cdot p(x_w^{t-1} \mid x_z^T) \quad (2)$$

and equals for $t = T+1$:

$$p(x_v^t \mid x_z^T) = a_{z,v} \quad (3)$$

Now consider the filter probability, i.e. the case where $T = t$. Exploiting the conditional independences $Y^t \perp Y^{1:t-1} \mid X^t$ and $X^t \perp Y^{1:t-1} \mid X^{t-1}$, and conditioning on X^{t-1}, we have the following relation between probabilities $p(x_v^t,\, y_e^{1:t})$ and $p(x_v^{t-1},\, y_e^{1:t-1})$ for two subsequent time slices $t-1$ and t, $t > 1$:

$$p(x_v^t,\, y_e^{1:t}) \;=\; o_{v,e_t} \cdot \sum_{z=1}^n a_{z,v} \cdot p(x_z^{t-1},\, y_e^{1:t-1}) \quad (4)$$

where e_t corresponds to the value of Y observed at time t. For time slice $t = 1$, we have that

$$p(x_v^1,\, y_e^1) = p(y_e^1 \mid x_v^1) \cdot p(x_v^1) = o_{v,e_1} \cdot \gamma_v \quad (5)$$

Note that for a prior marginal $p(x_i^t)$ we find the same expressions with o_{v,e_t} omitted.

Finally, consider the case where $T < t$. Equations 1 and 2 show that we now basically need to prolong the recursion in Equation 4 from time T to time t, except that for the time slices $T+1$ up to and including t no evidence is available. The absence of evidence can be implemented by multiplying with 1 rather than o_{v,e_t}. In the remainder of this section, we will therefore assume, without lack of generality, that $T = t$.

We will now translate the above relations into functions of the three types of model parameter. We already know that those functions are polynomial in the parameter under consideration, and we know the degree of the functions. However, we have yet to establish what the coefficients are and how to compute them. For ease of exposition concerning the covariation of parameters, we assume in the remainder of this section that all variables are binary-valued, i.e. $n = m = 2$.

3.2 Initial Parameters

We consider the sensitivity function $p(x_v^t,\, y_e^{1:t})(\theta_\gamma)$ for model parameter $\theta_\gamma = \gamma_r$. Note that γ_v, the parameter associated with the state of interest for X^t, corresponds either to θ_γ ($v = r$) or its complement $1-\theta_\gamma$ ($v \neq r$). From Equation 4 it now follows that for $t = 1$:

$$p(x_v^1,\, y_e^1)(\theta_\gamma) \;=\; \begin{cases} o_{v,e_1}\cdot\theta_\gamma + 0 & \text{if} \quad v = r \\ -o_{v,e_1}\cdot\theta_\gamma + o_{v,e_1} & \text{if} \quad v \neq r \end{cases}$$

and from Equation 5 we have for $t > 1$:

$$p(x_v^t,\, y_e^{1:t})(\theta_\gamma) = \;=\; \sum_{z=1}^{2} o_{v,e_t}\cdot a_{z,v}\cdot p(x_z^{t-1},\, y^{1:t-1})(\theta_\gamma)$$

The polynomial $p(x_v^t,\, y_e^{1:t})(\theta_\gamma)$ requires two coefficients: $c_{v,1}^t$ and $c_{v,0}^t$. Since each initial parameter is used only in time step 1, as the above expressions demonstrate, the coefficients for $t > 1$ can be established through a simple recursion for each $N = 0, 1$:

$$c_{v,N}^t = \sum_{z=1}^{2} o_{v,e_t}\cdot a_{z,v}\cdot c_{z,N}^{t-1}$$

with $c_{v,0}^1 = 0$ if $v = r$, and o_{v,e_1} otherwise; in addition $c_{v,1}^1 = o_{v,e_1}$ if $v = r$, and $-o_{v,e_1}$ otherwise.

3.3 Transition Parameters

We consider the sensitivity function $p(x_v^t,\, y_e^{1:t})(\theta_a)$ for model parameter $\theta_a = a_{r,s}$. From Equations 4 and 5 it follows that for $t = 1$ we find a constant, $p(x_v^1,\, y_e^1)(\theta_a) = o_{v,e_1}\cdot\gamma_v$, and for $t > 1$,

$$\begin{aligned} p(x_v^t,\, y_e^{1:t})(\theta_a) = \qquad & (6) \\ \;=\; & o_{v,e_t}\cdot\sum_{z=1}^{n} a_{z,v}(\theta_a)\cdot p(x_z^{t-1},\, y_e^{1:t-1})(\theta_a) \\ \;=\; & o_{v,e_t}\cdot a_{r,v}(\theta_a)\cdot p(x_r^{t-1},\, y_e^{1:t-1})(\theta_a) + \\ & + o_{v,e_t}\cdot a_{\overline{r},v}(\theta_a)\cdot p(x_{\overline{r}}^{t-1},\, y_e^{1:t-1})(\theta_a) \end{aligned}$$

where $\overline{r}$ denotes the state of X other than r. In the above formula, $a_{r,v}(\theta_a)$ equals θ_a for $v = s$ and $1-\theta_a$ for $v \neq s$; $a_{\overline{r},v}$ is independent of θ_a.

The polynomial $p(x_v^t,\, y_e^{1:t})(\theta_a)$ requires t coefficients: $c_{v,N}^t$, $N = 0, \ldots, t-1$. To compute these coefficients, building upon Equation 6 above, we designed a procedure which constructs a set of matrices containing the coefficients of the polynomial sensitivity functions for each hidden state and each time slice. We call this procedure the *Coefficient-Matrix-Fill* procedure.

3.3.1 The Coefficient-Matrix-Fill Procedure

The Coefficient-Matrix-Fill procedure constructs a matrix for each time slice under consideration, and fills this matrix with the coefficients of the polynomial functions relevant for that time slice. In this section, we will detail the procedure for computing the coefficients of $p(x_v^t,\, y_e^{1:t})(\theta_a)$. In the sections following, we demonstrate how a similar procedure can be used to compute the coefficients given an observation parameter, and in case $T \neq t$.

The basic idea For each time slice $k = 1, \ldots, t$ we construct an $n \times k$ matrix F^k. A row i in F^k contains exactly the coefficients for the function $p(x_i^k,\, y_e^{1:k})(\theta_a)$, so the procedure in fact computes the coefficients for the sensitivity functions for *all* n hidden states and *all* time slices up to and including t. A column j in F^k contains all coefficients of the $(j-1)$th-order terms of the n polynomials. More specifically, entry $f_{i,j}^k$ equals the coefficient $c_{i,j-1}^k$ of the sensitivity function $p(x_i^k,\, y_e^{1:k})(\theta_a)$. The entries of matrix F^1 are set to their correct values in

the initialisation phase of the procedure. Matrices F^k for $k > 1$ are built solely from the entries in F^{k-1}, the transition matrix A and the observation matrix O.

Fill operations The recursive steps in the various formulas are implemented by transitioning from matrix F^k to F^{k+1} for $k \geq 1$. To illustrate this transition, consider an arbitrary $(k-1)$th-degree polynomial in θ, $p(\theta) = c_{k-1}{\cdot}\theta^{k-1} + \ldots + c_1{\cdot}\theta + c_0$, and let this polynomial be represented in row i of matrix F^k, i.e. $f^k_{i,\cdot} = (c_0, \ldots, c_{k-1})$. In transitioning from matrix F^k to F^{k+1}, three types of operation (or combinations thereof) can be applied to $p(\theta)$:

- summation with another polynomial $(-)p^*(\theta)$ of the same degree: this just requires summing the coefficients of the same order, i.e. summing entries with the same column number;
- multiplication with a constant d: the resulting polynomial is represented in row i of matrix F^{k+1} by $f^{k+1}_{i,\cdot} = (d \cdot c_0, \ldots, d \cdot c_{k-1}, 0)$. Note that F^{k+1} has an additional column $k+1$, which is unaffected by this operation.
- multiplication with θ: the resulting kth-degree polynomial is represented in row i of matrix F^{k+1} by $f^{k+1}_{i,\cdot} = (0, c_0, \ldots, c_{k-1})$. This operation basically amounts to shifting entries from F^k one column to the right.

Fill contents: initialisation Matrix F^1 is initialised by setting $f^1_{i,1} = o_{i,e_1} \cdot \gamma_i$ for $i = 1, 2$. Matrices $F^2, \ldots, F^t$ are initialised by filling them with zeroes.

Fill contents: $k = 2, \ldots, t$ We will now provide the details for filling matrix F^k, $k > 1$. Following Equation 6, position j in row i of matrix F^k, $f^k_{i,j}$, $k > 1$, is filled with:

if $i = s$ then for $1 < j < k$:

$$o_{i,e_k} \cdot (f^{k-1}_{r,j-1} + a_{\overline{r},i} \cdot f^{k-1}_{\overline{r},j})$$

if $i \neq s$ then for $1 < j < k$:

$$o_{i,e_k} \cdot (-f^{k-1}_{r,j-1} + f^{k-1}_{r,j} + a_{\overline{r},i} \cdot f^{k-1}_{\overline{r},j})$$

For $j = 1$, the general cases above are simplified by setting $f^{k-1}_{r,j-1} = 0$. This boundary condition captures that entries in the first column correspond to coefficients of the zero-order terms of the polynomials and can therefore never result from a multiplication with θ_a. Similarly, since the coefficients for column $j = k$, $k > 1$, can *only* result from multiplication by θ_a, we set $f^{k-1}_{\cdot,j} = 0$ in that case.

Complexity In each of the k steps, an $n \times k$ matrix is filled. This matrix contains the coefficients for the functions $p(x^k_i, y^{1:k}_e)(\theta_a)$ for all i, so the procedure computes the coefficients for the sensitivity functions for all hidden states and all time slices up to and including t. If we are interested in only one specific time slice t, then we can save space by storing only two matrices at all times. The runtime complexity for a straightforward implementation of the algorithm is $O(n^2 \cdot t^2)$, which is t times that of the forward-backward algorithm. This is due to the fact that per hidden state we need to compute k numbers per time step rather than one.

Example 1. Consider an HMM with binary-valued hidden state X and binary-valued evidence variable Y. Let $\Gamma = [0.20, 0.80]$ be the initial vector for X^1, and let transition matrix A and observation matrix O be as follows:

$$A = \begin{bmatrix} 0.95 & 0.05 \\ 0.15 & 0.85 \end{bmatrix} \text{ and } O = \begin{bmatrix} 0.75 & 0.25 \\ 0.90 & 0.10 \end{bmatrix}$$

Suppose we are interested in the sensitivity functions for the two states of X^3 as a function of parameter $\theta_a = a_{2,1} = p(x^t_1 \mid x^{t-1}_2) = 0.15$, for all $t > 1$. Suppose the following sequence of observations is obtained: y^1_2, y^2_1 and y^3_1. To compute the coefficients for the sensitivity functions, the following matrices are constructed by the Coefficient-Matrix-Fill procedure:

$$F^1 = \begin{bmatrix} o_{1,2} \cdot \gamma_1 \\ o_{2,2} \cdot \gamma_2 \end{bmatrix} = \begin{bmatrix} 0.25 \cdot 0.20 \\ 0.10 \cdot 0.80 \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.08 \end{bmatrix}$$

$$F^2 = \begin{bmatrix} o_{1,1} \cdot a_{1,1} \cdot f^1_{1,1} & o_{1,1} \cdot f^1_{2,1} \\ o_{2,1} \cdot (f^1_{2,1} + a_{1,2} \cdot f^1_{1,1}) & -o_{2,1} \cdot f^1_{2,1} \end{bmatrix}$$

$$= \begin{bmatrix} 0.75 \cdot 0.95 \cdot 0.05 & 0.75 \cdot 0.08 \\ 0.90 \cdot (0.08 + 0.05 \cdot 0.05) & -0.90 \cdot 0.08 \end{bmatrix}$$

$$= \begin{bmatrix} 0.03563 & 0.06 \\ 0.07425 & -0.072 \end{bmatrix} + \\ \hline 0.10988 \quad -0.012$$

and finally, $F^3 =$

$$= \begin{bmatrix} o_{1,1}\cdot a_{1,1}\cdot f^2_{1,1} & & \\ o_{2,1}\cdot(f^2_{2,1} + a_{1,2}\cdot f^2_{1,1}) & & \\ & o_{1,1}\cdot(f^2_{2,1} + a_{1,1}\cdot f^2_{1,2}) & \\ & o_{2,1}\cdot(-f^2_{2,1} + f^2_{2,2} + a_{1,2}\cdot f^2_{1,2}) & \\ & & o_{1,1}\cdot f^2_{2,2} \\ & & -o_{2,1}\cdot f^2_{2,2} \end{bmatrix}$$

$$= \begin{bmatrix} 0.02538 & 0.09844 & -0.054 \\ 0.06843 & -0.12893 & 0.0648 \end{bmatrix} + \\ \hline 0.09381 \quad -0.03049 \quad 0.0108$$

From which we can conclude, for example:

$$p(x_1^3 \mid y_e^{1:3})(\theta_a) = \frac{-0.054\cdot\theta_a^2 + 0.098\cdot\theta_a + 0.025}{0.011\cdot\theta_a^2 - 0.030\cdot\theta_a + 0.094}$$

and also,

$$p(x_2^2 \mid y_e^{1:2})(\theta_a) = \frac{-0.072\cdot\theta_a + 0.074}{-0.012\cdot\theta_a + 0.110}$$

Note that the coefficients for the probability of evidence function follow from summing the entries in each column of F^t (see e.g. F^2 and F^3 above). □

3.4 Observation Parameters

We consider the sensitivity function $p(x_v^t,\, y_e^{1:t})(\theta_a)$ for model parameter $\theta_o = o_{r,s}$. From Equation 4 it follows that for $t = 1$:

$$p(x_v^1,\, y_e^1)(\theta_o) = \tag{7}$$

$$= \begin{cases} o_{v,e_1}\cdot\gamma_v & \text{if} \quad v \neq r; \\ \theta_o\cdot\gamma_r & \text{if} \quad v = r \text{ and } e_1 = s; \\ (1-\theta_o)\cdot\gamma_r & \text{if} \quad v = r \text{ and } e_1 \neq s; \end{cases}$$

and from Equation 5 we have for $t > 1$:

$$p(x_v^t,\, y_e^{1:t})(\theta_o) = \\ = o_{v,e_t}(\theta_o)\cdot\sum_{z=1}^{2} a_{z,v}\cdot p(x_z^{t-1},\, y^{1:t-1})(\theta_o) \tag{8}$$

where $o_{v,e_t}(\theta_o)$ equals o_{v,e_t} for $v \neq r$, θ_o for $v = r$ and $e_t = s$, and $1 - \theta_o$ for $v = r$ and $e_t \neq s$.

The polynomial function $p(x_v^t,\, y_e^{1:t})(\theta_o)$ requires $t+1$ coefficients: $c^t_{v,N}$, $N = 0, \ldots, t$. We compute these coefficients, building upon the Equations 7 and 8, again using our Coefficient-Matrix-Fill procedure. The contents and size of the matrices differ from the case with transition parameters and are specified below.

Fill contents: initialisation F^1 is an $n \times 2$ matrix, initialised in accordance with Equation 7. All F^k, $k = 2, \ldots, t$, are $n \times (k+1)$ matrices and are initialised by filling them with zeroes.

Fill contents: $k = 2, \ldots, t$ Following Equation 8, position j in row i of matrix F^k, $f^k_{i,j}$, $k > 1$, is filled with the following for $j = 2, \ldots, k$:

$$\begin{cases} \sum_{z=1}^{2} a_{z,i}\cdot f^{k-1}_{z,j-1} & \text{if} \quad i = r,\ e_t = s; \\ \sum_{z=1}^{2} a_{z,i}\cdot(f^{k-1}_{z,j} - f^{k-1}_{z,j-1}) & \text{if} \quad i = r,\ e_t \neq s; \\ o_{\overline{r},e_k}\cdot\sum_{z=1}^{2} a_{z,\overline{r}}\cdot f^{k-1}_{z,j} & \text{if} \quad i \neq r; \end{cases}$$

For $j = 1$ and $j = k+1$ we again simplify the above formulas where necessary, to take into account boundary conditions.

4 Smoothing Coefficients

In this section we consider establishing the coefficients of the sensitivity function $p(x_v^t \mid y_e^{1:T})(\theta)$ for various model parameters θ, in the situation where $t < T$. We again focus only on $p(x_v^t,\, y_e^{1:T})(\theta)$.

4.1 Recursion for Smoothing

Recall from Section 2 that $p(x_v^t,\, y_e^{1:T}) = B(i,t)\cdot F(i,t)$, or

$$p(x_v^t,\, y_e^{1:T}) = p(y_e^{t+1:T} \mid x_v^t)\cdot p(x_v^t,\, y_e^{1:t}) \tag{9}$$

The second term in this product is again a filter probability, so we now further focus on the first term. By conditioning on X^{t+1} and exploiting independences (see for details (Russel & Norvig, 2003, chapter 15)), we have the following relation between probabilities $p(y_e^{t+1:T} \mid x_v^t)$ and $p(y_e^{t+2:T} \mid x_v^{t+1})$ for $t+1 < T$:

$$p(y_e^{t+1:T} \mid x_v^t) = \\ = \sum_{z=1}^{n} o_{z,e_{t+1}}\cdot a_{v,z}\cdot p(y_e^{t+2:T} \mid x_z^{t+1}) \tag{10}$$

For $t+1 = T$, this reduces to

$$p(y_e^{T:T} \mid x_v^{T-1}) \;=\; \sum_{z=1}^{n} o_{z,e_T} \cdot a_{v,z} \cdot 1 \qquad (11)$$

Again we translate the above relations into functions of the various model parameters. From Equations 10 and 11 it follows that the function $p(y_e^{t+1:T} \mid x_v^t)(\theta)$ is polynomial in each model parameter.[2] Moreover, from Equation 9 we have that the degree of $p(x_v^t, y_e^{1:T})(\theta)$ equals the sum of the degrees of $p(y_e^{t+1:T} \mid x_v^t)(\theta)$ and $p(x_v^t, y_e^{1:t})(\theta)$. Since the degrees of both $p(x_v^t, y_e^{1:T})(\theta)$ and $p(x_v^t, y_e^{1:t})(\theta)$ are known (see Section 2), the degree of $p(y_e^{t+1:T} \mid x_v^t)(\theta)$ can be established as their difference. We thus have that

$$p(y_e^{t+1:T} \mid x_v^t)(\theta) = d_{v,N}^t \cdot \theta^N + \ldots + d_{v,1}^t \cdot \theta + d_{v,0}^t$$

where

$$N = \begin{cases} T-t & \text{if} \quad \theta = o_{r,s} \text{ or } \theta = a_{r,s}; \\ 0 & \text{if} \quad \theta = \gamma_r \end{cases}$$

and coefficients $d_{v,N}^t, \ldots, d_{v,0}^t$ are constants with respect to the various parameters. The coefficients of the polynomial function $p(x_v^t, y_e^{1:T})(\theta)$, $T > t$, can thus be established by standard polynomial multiplication of $p(x_v^t, y_e^{1:t})(\theta)$ and $p(y_e^{t+1:T} \mid x_v^t)(\theta)$.

In the following we will establish exactly what the coefficients of $p(y_e^{t+1:T} \mid x_v^t)(\theta)$ are and how to compute them. For ease of exposition, we again take $n = m = 2$.

4.2 Initial Parameters

We consider the function $p(y_e^{t+1:T} \mid x_v^t)(\theta_\gamma)$, $t < T$, for model parameter $\theta_\gamma = \gamma_r$. The degree of this polynomial is 0. Indeed, from Equations 10 and 11 it follows that this function is constant with respect to an initial parameter. This constant is simply a probability which can be computed using standard inference.

4.3 Transition Parameters

The function $p(y_e^{t+1:T} \mid x_v^t)(\theta_a)$, $t < T$, with parameter $\theta_a = a_{r,s}$ requires $T - t + 1$ coefficients. We again compute these coefficients using our Coefficient-Matrix-Fill procedure, where contents is now determined by Equations 10 and 11, and depends on the relation between $a_{v,z}$ and θ_a: if $v = r$ then $a_{v,z}$ equals θ_a for $z = s$, and $1 - \theta_a$ for $z = \bar{s}$; otherwise $a_{v,z}$ is constant.

To distinguish between computations that move forward in time, and the current ones which move backward in time, we will use matrices B^k, $t \leq k \leq T$, where $k = T$ is used purely as initialisation.

Fill contents: initialisation B^T is an $n \times 1$ matrix, initialised with 1's. All B^k, $k = t, \ldots, T-1$, are $n \times (T - k + 1)$ matrices which are initialised with zeroes.

Fill contents: $k = T-1$ down to t Following Equations 10 and 11, position j in row i of matrix B^k, $b_{i,j}^k$, $k < T$, is filled with the following for $j = 2, \ldots, T-k$:

if $i = r$**:** $\left(\sum_{z=1}^{2} o_{z,e_{k+1}} \cdot b_{z,j-1}^{k+1}\right) + o_{\bar{s},e_{k+1}} \cdot b_{\bar{s},j}^{k+1}$

if $i \neq r$**:** $\sum_{z=1}^{2} o_{z,e_{k+1}} \cdot a_{i,z} \cdot b_{z,j}^{k+1}$

For $j = 1$ and $j = T - k + 1$ we again have to take into account boundary conditions.

4.4 Observation Parameters

The function $p(y_e^{t+1:T} \mid x_v^t)(\theta_o)$, $t < T$, with parameter $\theta_o = o_{r,s}$ requires $T - t + 1$ coefficients. We again compute these coefficients using our Coefficient-Matrix-Fill procedure in a similar way as for the transition parameters above. The only difference is in the fill contents determined by Equations 10 and 11. This now depends on the relation between $o_{z,e_{t+1}}$ and θ_o: for $z = r$, $o_{z,e_{t+1}}$ equals θ_o if $e_{t+1} = s$, and $1 - \theta_a$ if $e_{t+1} \neq s$; for $z = \bar{r}$, $o_{z,e_{t+1}}$ is constant.

Fill contents: $k = T-1$ down to t Following Equations 10 and 11, position j in row i of matrix B^k, $b_{i,j}^k$, $k < T$, is filled with the following for $j = 2, \ldots, T-k$:

if $e_{k+1} = s$**:** $a_{i,r} \cdot b_{r,j-1}^{k+1} + o_{\bar{r},e_{k+1}} \cdot a_{i,\bar{r}} \cdot b_{\bar{r},j}^{k+1}$

if $e_{k+1} \neq s$**:** $-a_{i,r} \cdot b_{r,j-1}^{k+1} + a_{i,r} \cdot b_{r,j}^{k+1} + o_{\bar{r},e_{k+1}} \cdot a_{i,\bar{r}} \cdot b_{\bar{r},j}^{k+1}$

For $j = 1$ and $j = T - k + 1$ we again take into account the boundary conditions.

[2] Note that this may seem counter-intuitive as it concerns the function for a *conditional* probability; since X^t is an ancestor of $Y^{t+1} \ldots Y^T$, however, the factorisation of $p(y_e^{t+1:T}, x_v^t)$ includes $p(x_v^t)$.

5 Related Work

Varying a transition or observation parameter in an HMM corresponds to varying multiple parameters in its Bayesian network representation, one for each time slice under consideration. Sensitivity analysis in HMMs is therefore a constrained form of n-way analysis in Bayesian networks, with all varied parameters having the same value at all times. As a result, a sensitivity function in an HMM requires a number of coefficients linear in the number of parameters varied, whereas in Bayesian networks in general an n-way sensitivity function requires an exponential number of coefficients, one for each possible subset of the n varied parameters. For Bayesian networks, n-way sensitivity analysis, with parameters from *different* CPTs, has been studied by only few (see Coupé et al. (2000) for an overview and comparison of research). For computing the coefficients of n-way sensitivity functions roughly three approaches, or combinations thereof, are known: symbolic propagation, solving systems of linear equations, and propagation of tables with coefficients. The approach taken by Coupé et al. (2000) resembles our Coefficient-Matrix-Fill procedure in the sense that a table or matrix of coefficients is constructed; their approach extends the junction-tree architecture to propagate vector tables rather than potential functions and defines operations on vectors to this end. Each vector table contains the coefficients of the corresponding potential function is terms of the parameters under study. Our approach, on the contrary, does not depend on a specific computational architecture nor does it necessarily require a Bayesian network representation of the HMM. In addition, the operations we use are quite different, since we can exploit the fact that we have a polynomial function in a single parameter.

6 Conclusions and Further Research

In this paper we introduced a new and efficient algorithm for computing the coefficients of sensitivity functions in Hidden Markov Models, for all three types of model parameter. Earlier work on this topic suggested to use the Bayesian network representation of HMMs and associated algorithms for sensitivity analysis. In this paper we have shown that exploiting the repetitive character of HMMs results in a simple algorithm that computes the coefficients of the sensitivity functions for all hidden states and all time steps. Our procedure basically mimics the forward-backward inference algorithm, but computes coefficients rather than probabilities. Various improvements of the forward-backward algorithm for HMMs exist that exploit the matrix formulation (Russel & Norvig, 2003, Section 15.3); further research is required to investigate if our procedure can be improved in similar or different ways.

The presented work can be extended quite straightforwardly to sensitivity functions which concern the prediction of future observations, i.e $p(y_e^t \mid y_e^{1:T})(\theta)$, $T < t$. More challenging will be to extend current research to sensitivity analysis in which different types of model parameter are varied simultaneously, and to extensions of HMMs.

References

Th. Charitos, L.C. van der Gaag (2004). Sensitivity properties of Markovian models. *Proceedings of Advances in Intelligent Systems - Theory and Applications Conference (AISTA)*. IEEE Computer Society.

Th. Charitos (2007). *Reasoning with Dynamic Networks in Practice*. PhD Thesis, Utrecht University, The Netherlands.

V.M.H. Coupé, F.V. Jensen, U. Kjærulff, L.C. van der Gaag (2000). *A computational architecture for n-way sensitivity analysis of Bayesian networks.* Technical Report: Department of Computer Science, Aalborg University

L.C. van der Gaag, S. Renooij, V.M.H. Coupé (2007). Sensitivity analysis of probabilistic networks. In: *Advances in Probabilistic Graphical Models*, Springer Series: Studies in Fuzziness and Soft Computing, Vol. 213, pp. 103-124.

A.Yu. Mitrophanov, A. Lomsadze, M. Borodovsky (2005). Sensitivity of hidden Markov models. *Journal of Applied Probability*, 42, pp. 632–642.

K.P. Murphy (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD Thesis, University of California, Berkeley.

S. Russel, P. Norvig (2003) *Artificial Intelligence: A Modern Approach*, Prentice Hall, Second Edition.

P. Smyth, D. Heckerman, M.I. Jordan (1997). Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9, pp. 227–269.

Learning undirected graphical models from multiple datasets with the generalized non-rejection rate

Alberto Roverato
Università di Bologna, Italy
alberto.roverato@unibo.it

Robert Castelo
Universitat Pompeu Fabra, Barcelona, Spain
robert.castelo@upf.edu

Abstract

Learning graphical models from multiple datasets constitutes an appealing approach to learn transcriptional regulatory interactions from microarray data in the field of molecular biology. This has been approached both in a model based statistical approach and in an unsupervised machine learning approach where, in the latter, it is common practice to pool datasets produced under different experimental conditions. In this paper, we introduce a quantity called the *generalized non-rejection rate* which extends the non-rejection rate, introduced by Castelo and Roverato (2006), so as to explicitly keep into account the different graphical models representing distinct experimental conditions involved in the structure of the dataset produced in multiple experimental batches. We show that the generalized non-rejection rate allows one to learn the common edges occurring throughout all graphical models, making it specially suited to identify robust transcriptional interactions which are common to all the considered experiments. The generalized non-rejection rate is then applied to both synthetic and real data and shown to provide competitive performance with respect to other widely used methods.

1 Introduction

In the field of molecular biology, an important process that takes place within the cell is gene expression, where the DNA sequence of a gene is transcribed into a functional RNA molecule which, in the case of protein-coding genes, is translated into a protein. When, where and how often a gene is expressed is determined by the requirements imposed by the cell in order to fulfill the corresponding cellular functions. The control exerted on the expression of every gene is known as gene regulation and takes place through a wide range of mechanisms acting at different levels of the gene expression pathway in a coordinated manner. One such mechanisms is the initiation of the synthesis of the RNA molecule, or transcription initiation, which, among other things, requires the presence of a specific combination of proteins that belong to a class of genes called transcription factors. Transcription factor proteins play their role in the initiation of transcription by binding to the upstream genomic region of the regulated gene and then promoting the initiation of transcription (up-regulation) or repressing that step (down-regulation).

The interactions between transcription factors and the genes they target under specific cell environmental conditions, constitute one of the key pieces of information in the cellular program that governs gene expression. Therefore, identifying transcriptional regulatory interactions is a fundamental step towards reverse-engineering this cellular program which potentially contains clues to understanding biological processes like cell division, cell fate or disease.

Microarray technology in molecular biology enables measuring gene expression simultaneously for thousands of genes across a moderate number of samples corresponding to technical or biological replicates of one or more distinct experimental con-

Pp. 249–257 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

ditions. The resulting gene expression data matrix conveys a snapshot of the expression level of genes under the essayed experimental conditions and efforts have been made in the last years in order to develop computational and statistical procedures that aid to infer, from these data, transcriptional regulatory interactions suitable of being followed-up by further functional experimental validation. Among those procedures, we shall distinguish in this paper between *model based statistical learning methods* and *unsupervised machine learning methods*. In unsupervised machine learning it is common practice to apply the learning algorithm to pooled datasets constructed by merging samples from smaller datasets generated under different experimental conditions (Wang et al., 2006; Steele and Tucker, 2008). The performance of different learning algorithms is then assessed with respect to benchmark problems for which the set of regulatory interactions between genes is (partially) known and available from the biological literature.

Castelo and Roverato (2006) introduced a quantity that they called the *non-rejection rate* and used it in the model based learning of transcriptional networks. Furthermore, Castelo and Roverato (2009) showed that the non-rejection rate provides satisfying results also in an unsupervised machine learning approach. In this paper, we introduce a generalized version of the non-rejection rate that can be naturally used as a meta-analysis approach when the available data are a compendium of microarray experiments. We show that it is suited for the unsupervised machine learning of robust transcriptional interactions, that is transcriptional interactions that are common to all the experimental conditions considered. We apply the proposed method to synthetic and experimental data from one of the best characterized organisms in terms of its transcriptional regulatory relationships, the bacterium Escherichia coli (E. coli), and compare it with some widely used unsupervised learning procedures.

The paper is organized as follows. In Section 2 we review the theory related to structural learning of biological networks. Section 3 addresses the issue of meta-analysis and introduces the generalized non-rejection rate. In Section 4 an analysis based on simulated data is carried out whereas in Section 5 the generalized non-rejection rate is compared with other unsupervised learning procedures on a microarray dataset from the E. coli system. Finally, Section 6 contains a brief discussion.

2 Background

2.1 Unsupervised machine learning of biological networks

There is a substantial amount of recent work on high-dimensional and computationally tractable procedures for learning of biological networks. These can be grouped into two main families: *model based* and *unsupervised machine learning* procedures.

Model based procedures mainly rely on graphical models (see, for instance, Friedman, 2004). Graphical models are well-defined statistical models and the associated network has a precise probabilistic interpretation in terms of conditional independencies. Several procedures are available for learning an independence graph from data (Edwards, 2000), however, structural learning in this context poses new challenges because in microarray data the sample size n is smaller than the number of variables p. This has led to the development of specific structural learning procedures which try to overcome the *small n and large p problem* by exploiting specific biological background knowledge on the structure of the network. From this viewpoint, the most relevant feature of biological networks is that they are *sparse*, that is the direct regulatory interactions between genes represent a small proportion of all possible interactions in the network (see Junker, 2008).

With the term *unsupervised machine learning approach*, shortly *unsupervised learning*, we mean a set of methods that distinguish themselves from the model based statistical learning procedures both for the assumptions underlying the analysis and for the interpretation of the results (see d'Alché-Buc, 2006). More specifically, unsupervised learning procedures aim at identifying some "direct", to be read as "non-spurious", associations with high confidence (see, for instance, Faith et al., 2007) and, usually, no underlying statistical model is assumed. In this framework, the most popular procedures belong to the family of *relevance networks*. One of the first applications of relevance networks was by Butte and Kohane (2003) who proposed to consider

some pairwise measure of association between two expression profiles (e.g., Pearson correlation coefficient or mutual information), compute it for every pair of genes of interest (e.g., transcription-factor gene vs. target gene) and output those gene pairs with an association strength above a given threshold. Widely used enhancements to this pairwise approach aimed at reducing the number of identified spurious associations are the ARACNE procedure (Margolin et al., 2006) and the CLR procedure (Faith et al., 2007).

In order to validate and compare different unsupervised learning procedures it may be useful to make use of a *benchmark set of transcriptional interactions*, hereafter *benchmark set* for short, that can be obtained by mining existing literature on functional experiments that essay the actual activation or inhibition of a gene by a transcription factor. The benchmark set can be used to construct several measures of performance of the procedure. In particular, specificity and sensitivity can be used to produce ROC curves; however, because of the sparsity of biological networks, and of the consequent large skew in the class distribution, ROC curves present an overly optimistic view of an algorithm's performance (see Fawcett, 2006). In this case, ROC analyses are usually replaced by a *precision-recall curve* where the fraction of the interactions in the benchmark set that the procedure successfully identifies (*recall*) is plotted against the fraction of identified interactions that are true positives (*precision*). We also provide a summary value associated to a precision-recall curve obtained by computing its Area Under the Curve (AUC).

The connection between the model based and the unsupervised learning approaches comes from the fact that model based learning procedures which provide a measure of association of the edges of the complete graph can also be applied in unsupervised learning; see, for instance, Soranzo et al. (2007). This is the case, for instance, of the shrinkage estimator (Schäfer and Strimmer, 2005) implemented in the R package GeneNet. Furthermore, Castelo and Roverato (2006) introduced a quantity that they called the non-rejection rate to be applied in a model based approach but they then showed (Castelo and Roverato, 2009) that it provides satisfying results also in an unsupervised learning approach.

2.2 The non-rejection rate

For a finite set $V = \{1, \ldots, p\}$ let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. Furthermore, let X_V be a multivariate normal random vector, indexed by V, with mean vector μ and covariance matrix Σ. The probability distribution of X_V is said to belong to a Gaussian graphical model with graph G if every missing edge in the graph, $(i, j) \notin E$, implies that X_i is conditionally independent of X_j given the remaining variables $X_{V\setminus\{i,j\}}$. We refer to Lauritzen (1996) for a comprehensive account on Gaussian graphical models, but it is worth recalling here that such conditional independence relationship holds if and only if the partial correlation coefficient $\rho_{ij.V\setminus\{i,j\}}$ is equal to zero.

Led D be a random sample of n observations i.i.d. from X_V. The non-rejection rate is a quantity, introduced by Castelo and Roverato (2006), that can be used in structural learning of Gaussian graphical models when $p > n$ and the network has a sparse structure. It is based on *q-order partial correlations*, that is on partial correlations $\rho_{ij.Q}$ where $Q \subset V\setminus\{i, j\}$ is such that $|Q| = q$.

In the rest of this section we review the theory of the non-rejection rate required for this paper and refer to Castelo and Roverato (2006) for a more complete discussion. The non-rejection rate is a measure associated with every pair of vertices $i, j \in V$ and depends on the particular value of $q < (n - 2)$ being used. It can be described as follows: let $\mathcal{Q}_{ij}$ be the set made up of all subsets $Q \in V\setminus\{i, j\}$ such that $|Q| = q$. Let T_{ij} be a binary random variable associated to the pair of vertices i and j that takes values from the following two-stage experiment:

1. An element Q is sampled from $\mathcal{Q}_{ij}$ according to a (discrete) uniform distribution;

2. using the available data D the null hypothesis $H_0 : \rho_{ij.Q} = 0$ is verified on the basis of a statistical test of level α. If H_0 is rejected then T_{ij} takes value 0, otherwise it takes value 1.

Thus T_{ij} is a Bernoulli random variable and the non-rejection rate is defined as its expectancy; formally

$$\text{NRR}(i, j \mid q, D) := E[T_{ij}] = \Pr(T_{ij} = 1).$$

It is shown in Castelo and Roverato (2006) that the non-rejection rate can be written as

$$\begin{aligned} \text{NRR}(i,j\,|q,D) &= \beta_{ij}(1-\pi_{ij})+(1-\alpha)\pi_{ij} \\ &= \beta_{ij}+(1-\alpha-\beta_{ij})\pi_{ij} \quad (1) \end{aligned}$$

where α is the probability of the first type error of the used statistical test, π_{ij} is the proportion of subsets Q in $\mathcal{Q}_{ij}$ that separate i and j in G and β_{ij} is the mean value of the second type errors $\beta_{ij.Q}$ for all the subsets $Q \in \mathcal{Q}_{ij}$ such that Q does not separate i and j in G; see Lauritzen (1996, p. 6) for a formal definition of separator. It follows that the non-rejection rate is a probability that takes a value between 0 and $(1-\alpha)$. If a pair of vertices i and j are adjacent, that is $(i,j) \in G$, (in our context this is as much as saying that a transcription factor directly binds to the promoter region of a target gene) then $\pi_{ij} = 0$ and equation (1) makes clear that the theoretical non-rejection rate equals exactly β_{ij}. Hence, since the statistical power of the tests for zero partial correlation with null hypothesis $H_0 : \rho_{ij.Q} = 0$ equals $1-\beta_{ij.Q}$, then the non-rejection rate of an edge $(i,j) \in G$ corresponds to the one minus the average statistical power to detect that association. It follows that for $(i,j) \in G$ the non-rejection rate $\text{NRR}(i,j\,|q,D)$ converges to 0 as $n-q$ goes to infinity whereas for finite sample size it is a summary measure of the strength of the linear association represented by the edge (i,j) over all the marginal distributions of size $q+2$, with 0 representing maximum strength.

3 Meta-analysis

In unsupervised learning it is common practice to overcome the difficulties related to the small sample size by applying the procedures in a meta-analysis approach. More specifically, a pooled dataset is obtained by merging smaller datasets generated under different experimental conditions. We remark that this practice would make little sense in a model based approach, whereas in a unsupervised learning approach it is justified whenever it leads to an improvement of the precision-recall performance of the procedure.

In this section we formally approach this issue and assume that the data involve a common set of genes V but are obtained from m batches of microarray experiments, possibly under different experimental conditions. Formally, set $M = \{1,\ldots,m\}$ and for every microarray experiment $s \in M$ let $X_V^{(s)}$ be a random vector, indexed by V, corresponding to the expression level of genes in the experimental condition $s \in M$. Furthermore, let $D^{(s)}$ be a random sample of $n^{(s)}$ i.i.d. observations from $X_V^{(s)}$ so that $D^* = \{D^{(1)},\ldots,D^{(m)}\}$ is the pooled dataset made up of $n = \sum_{s=1}^m n^{(s)}$ independent, but not identically distributed, observations.

Our standpoint is that biological networks are dynamic objects which modify their interaction structure to allow the cell to respond effectively to changes of its internal and external environments. This is formalized by assuming that a different graph is associated with every experimental condition and, specifically, by assuming that for every $s \in M$ the probability distribution of $X_V^{(s)}$ belongs to an undirected Gaussian graphical model with graph $G^{(s)} = (V, E^{(s)})$.

In the following section we introduce the generalized non-rejection rate which keeps into explicit account the pooled structure of the dataset.

3.1 The generalized non-rejection rate

The non-rejection rate is defined in Section 2.2 with respect to a set D of i.i.d. observations as the expected value of a Bernoulli random variable generated by means of a two stage experiment. Similarly, the *generalized non-rejection rate* between two variables, X_i and X_j is defined with respect to the dataset D^* as the expected value of a Bernoulli random variable T_{ij}^*,

$$\text{gNRR}(i,j\,|q,D^*) := E[T_{ij}^*] = \Pr(T_{ij}^* = 1),$$

where T_{ij}^* is defined by adding a third step to the two stage experiment as follows:

1. A random value s is generated from S, where S is a discrete random variable that takes values in M with probability $\Pr(S = s) = n^{(s)}/n$;
2. an element Q is sampled from $\mathcal{Q}_{ij}$ according to a (discrete) uniform distribution;
3. using the dataset $D^{(s)}$ the null hypothesis $H_0 : \rho_{ij.Q} = 0$ is verified on the basis of a statistical

test of level α. If H_0 is rejected then T^*_{ij} takes value 0, otherwise it takes value 1.

The value q is chosen so that $q < \min\{(n^{(s)} - 2); s \in M\}$.

We use the generalized non-rejection rate values to produce a ranking of all possible pair of genes, i.e., of all possible edges of the graph, and its usefulness will be assessed by means of a precision-recall analysis; nevertheless, in the following we shortly discuss the interpretation of this quantity.

As well as the non-rejection rate, also the generalized non-rejection rate is a probability. Furthermore, it can be written as weighted average of the non-rejection rates for the single datasets with weights proportional to sample sizes,

$$\begin{aligned}\text{gNRR}(i,j\,|q,D^*) &= \sum_{s=1}^{m} \text{NRR}(i,j\,|q,D^{(s)})P(S=s)\\ &= \frac{1}{n}\sum_{s=1}^{m} \text{NRR}(i,j\,|q,D^{(s)})\,n^{(s)}.\end{aligned}$$

It follows that if $(i,j) \in G^{(s)}$ for every $s \in M$, then the generalized non-rejection rate is the weighted average of the mean second type errors $\beta^{(s)}_{ij}$ for all the datasets $D^{(s)} \in D^*$, formally $\text{gNRR}(i,j\,|q,D^*) = \bar{\beta}_{ij}$ where $\bar{\beta}_{ij} := \sum_{s=1}^{m} \beta^{(s)}_{ij} P(S=s)$. Since for all $s \in M$ the quantity $\beta^{(s)}_{ij}$ converges to zero as $n^{(s)}$ increases, it follows that the generalized non-rejection rate is specially useful to identify edges that belong to all graphs. These are robust transcriptional interactions that are common to all the experiments considered.

4 Assessment with simulated data

In order to empirically show the behavior of the generalized non-rejection rate we considered $m = 2$ and two different graphs $G^{(1)}$ and $G^{(2)}$ and repeat the following simulation 100 times.

1. Generate randomly $G^{(1)}$ and $G^{(2)}$ with $p = 50$ vertices each of them with an average number of adjacent vertices equal to 3.

   For each of these two graphs build a random precision matrix whose zero structure reflects the conditional independencies encoded in the graph (i.e., the pattern of missing edges) and sample $n = 30$ observations from the Gaussian distribution with that precision matrix and zero mean vector. This step provides us with two datasets, $D^{(1)}$ and $D^{(2)}$, with $p = 50$ and $n = 30$ belonging to two different joint multivariate Gaussian distributions.

2. Estimate the non-rejection rate with $q = 4$ and the Pearson correlation coefficient for each pair of variables, separately from $D^{(1)}$ and $D^{(2)}$. Apply methods GeneNet, ARACNE and CLR separately to $D^{(1)}$ and $D^{(2)}$.

3. Build the collection of datasets $D^* = \{D^{(1)}, D^{(2)}\}$. By using D^* estimate the generalized non-rejection rate, the non-rejection rate and the Pearson correlation for each pair of variables. Apply methods GeneNet, ARACNE and CLR to D^*.

4. For each method and dataset, calculate a precision-recall curve with respect to the union graph $G^* = (V, E^{(1)} \cup E^{(2)})$. For each precision-recall curve calculate the area under this curve (AUC).

5. In order to have a baseline comparison method we have generated three sets of random correlations sampling values uniformly from (-1,+1) for each pair of the $p = 50$ variables. This will be referred to as the Random method.

In Figure 1 we show the AUC values across the 100 simulations for each of the methodologies followed where the larger the value, the better the performance. The two panels on top correspond to using $D^{(1)}$ and $D^{(2)}$ separately in order to try to infer all the edges in the union of the two graphs. The panel at the bottom shows the performance when using a meta-analysis approach in which all datasets in D^* are used together. We can observe that only the generalized non-rejection rate provides a clear improvement over the use of any of the methods on one single dataset. GeneNet also achieves a small increase in its median AUC value.

We have previously pointed out that the generalized non-rejection rate is specially useful to identify edges that belong to all graphs. In the previous simulations those correspond to edges in both $G^{(1)}$ and $G^{(2)}$ in each simulation. We have used the previous

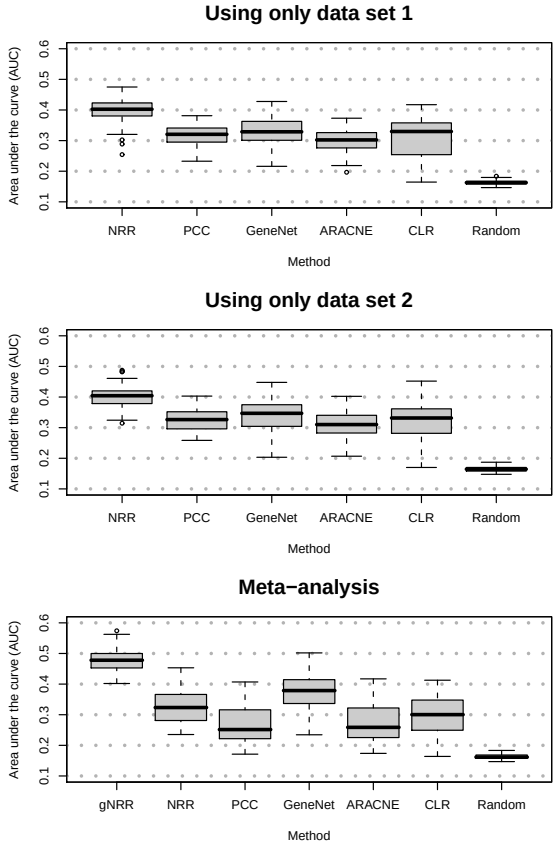


Figure 1: AUC value comparison (the larger the value the better the performance).

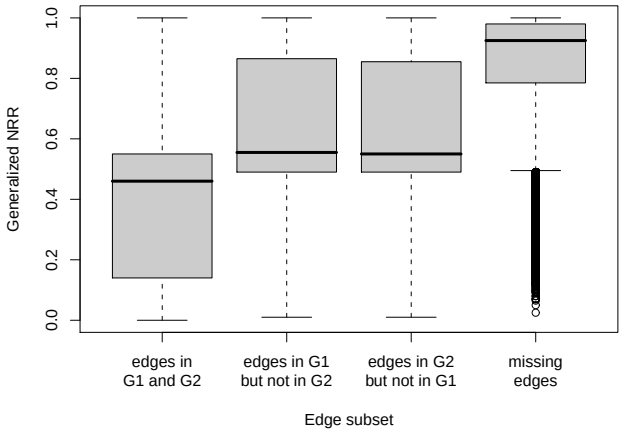


Figure 2: Behaviour of the generalized non-rejection rate for different kind of edges in graphs $G^{(1)}$ and $G^{(2)}$.

results to empirically verify this fact by grouping together the generalized non-rejection rate values in the following four subsets: edges in $G^{(1)}$ and $G^{(2)}$, edges in $G^{(1)}$ but not in $G^{(2)}$, edges in $G^{(2)}$ but not in $G^{(1)}$ and missing edges in both $G^{(1)}$ and $G^{(2)}$. Then, we have examined the distribution of generalized non-rejection rates separately for each of these subsets and we may see those values from the 100 simulations in Figure 2. Clearly, generalized non-rejection rates are most of the time smaller for edges that belong to the two graphs than for edges that belong to one of the two graphs only or edges that are missing in both graphs.

5 Assessment with microarray data

In order to assess with real microarray experimental data whether the generalized non-rejection rate increases our accuracy when trying to identify transcriptional regulatory relationships through a meta-analysis approach, we have used experimental and functional annotation data from the E. coli system. These bacteria are the free-living organism for which the largest part of its transcriptional regulatory network has undergone some sort of experimental validation. The database RegulonDB (Gamma-Castro et al., 2008) contains a set of transcription factor and target gene relationships curated from the literature, and we have used those as benchmark set of regulatory interactions. The E. coli microarray dataset that we have used to assess our meta-analysis approach corresponds to the oxygen deprivation data from Covert, et al. (2004), available from the Gene Expression Omnibus (GEO) database (Barrett, 2007) with accession number GDS680, which monitors the response of E. coli during the transition from aerobic to anaerobic conditions which are essayed in two groups of 21 and 22 experiments, respectively. In order to obtain the necessary variability on the expression levels of genes that form part of the transcriptional network relevant to these experiments, Covert, et al. (2004) used six E. coli strains with knockouts of key transcriptional regulators in the oxygen response (Δ*arcA*, Δ*appY*, Δ*fnr*, Δ*oxyR*, Δ*soxS* and the double knockout Δ*arcA*Δ*fnr*). Both, the microarray oxygen deprivation data where we are going to assess our approach and the RegulonDB interactions which we are going to use as

benchmark set, were pre-processed by Castelo and Roverato (2009) and are available as part of the `qpgraph` package from the Bioconductor project website (`http://www.bioconductor.org`). These pre-processed datasets consist, in one hand, of 3283 transcriptional regulatory interactions in RegulonDB involving 1428 genes and, on the other hand, $p = 4205$ genes and $n = 43$ experiments in the oxygen deprivation microarray expression data matrix. From these latter dataset, we have discarded one of the aerobic experiments (GSM18237) which showed a very dissimilar profile to the rest of the aerobic experiments. More concretely, we have used only those genes involved in the regulatory modules of the five transcription factors knocked-out in the experiments of Covert, et al. (2004) according to the RegulonDB database. This subnetwork is formed by 378 genes out of which 22 are transcription factors involved in 681 transcriptional interactions. In the bottom panel of Figure 3 we may see the resulting precision-recall curves where we compare the generalized non-rejection rate with $q = 10$ (gNRR) with other methods applied to the union of the aerobic and anaerobic datasets including the non-rejection rate with $q = 15$ (NRR), the absolute Pearson correlation coefficient (PCC), ARACNE, CLR, GeneNet and the assignment of a uniformly random correlation between every transcription factor and target gene (Random). The gNRR improves the rest of the meta-analysis approaches up to a 40% larger AUC with respect to the second best-performing approach (NRR).

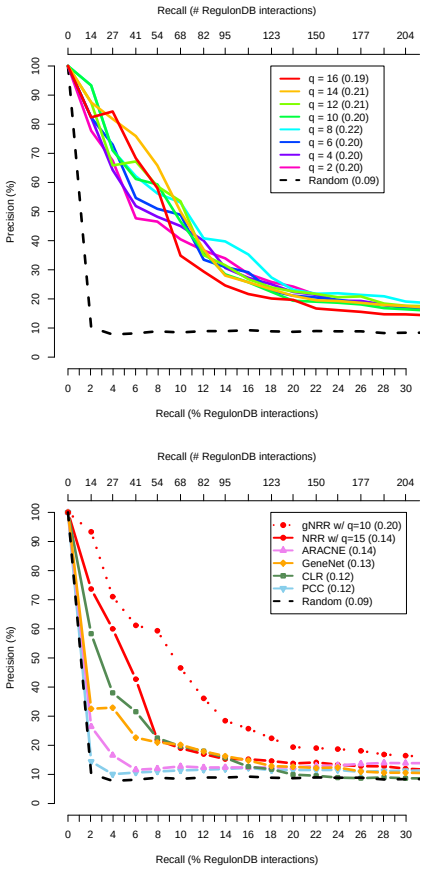


Figure 3: Comparison of precision-recall curves through the first 30% fraction of the recall. The top panel shows curves from the generalized non-rejection rate for a wide range of possible q values. The bottom panel shows curves for different unsupervised machine learning methods. The AUC value is the number enclosed within brackets.

An issue in the computation of the non-rejection rate concerns the choice of the parameter q. In this application, q may take any value between 1 and 18 and there is a trade-off between larger values of q, which increase the probabilities π_{ij}, and smaller values of q, which improve the power of statistical tests. In the bottom panel of Figure 3 we set $q = 10$ for gNRR which is the median of the possible values of q within a sensible range bounded by the number of available samples but, in fact, our procedure is not very sensitive to the choice of q as shown in the top panel of Figure 3 where the precision-recall curves of gNRR for different values of q are drawn.

6 Discussion

The idea at the basis of the generalized non-rejection rate is that of exploiting the information provided by the different microarray experiments on the common part of the underlying regulatory networks in a way that does not require pooling the datasets. We believe that this is the crucial aspect that makes the generalized non-rejection rate outperform other procedures, but it is worth pointing out that it also has the drawback that the power and efficiency of the applied statistical procedures depend on the samples sizes of the single experiments, $n^{(s)}$, rather than the overall sample size n.

Acknowledgments

This work is supported by the Spanish Ministerio de Ciencia e Innovación (MICINN) [TIN2008-00556/TIN] and the ISCIII COMBIOMED Network [RD07/0067/0001]. The first author acknowledges support from the Ministero dell'Università e della Ricerca [PRIN-2007AYHZWC, FISR MITICA]. Robert Castelo is a research fellow of the "Ramon y Cajal" program from the Spanish MICINN [RYC-2006-000932].

References

Barrett, T., Troup, D.B., Wilhite, S.E., Ledoux, P., Rudnev, D. Evangelista, C. Kim, I.F., Soboleva, A., Tomashevsky, M. and Edgar, R. 2007. NCBI GEO: mining tens of millions of expression profiles–database and tools update. Nucleic Acids Research, 35 D760-D765.

Butte, A.S. and Kohane,I.S., 2003. Relevance networks: a first step toward finding genetic regulatory networks within microarray data. In G. Parmigiani, E.S. Garett, R.A. Irizarry and S.L. Zeger (Ed.s), The Analysis of Gene Expression Data, Springer, New York, 428-446.

Castelo, R. and Roverato, A., 2006. Gaussian graphical model search from microarray data with p larger than n. Journal of Machine Learning Research, 7 2621-2650.

Castelo, R. and Roverato, A., 2009. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. Journal of Computational Biology, 16(2):213-227.

Covert, M.W., Knight, E.M., Reed, J.L., Herrgard, M.J. and Palsson, B.O. 2004. Integrating high-throughput and computational data elucidates bacterial networks. Nature, 429 92-96.

d'Alché-Buc, F., 2006. Inference of biological regulatory networks: machine learning approaches. In: F. Képès, (Ed.). Biological networks, World Scientific, 41-82.

Edwards, D.E., 2000. Introduction to graphical modelling. Springer-Verlag, New York.

Fawcett, T., 2006. An introduction to ROC analysis. Pattern recognition letters, 27 861-874.

Faith, J.J., Hayete, B., Thaden, J.T., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J.J. and Gardner, T.S., 2007. Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. PLoS Biol, 5(1) e8.

Friedman. N., 2004. Inferring cellular network using probabilistic graphical models. Science, 33 799-805.

Junker, B.H., 2008. Networks in biology. In: B.H. Junker and F. Schreiber (Ed.s). Analysis of Biological Networks, Wiley, 3-12.

Lauritzen, S.L., 1996. Graphical models. Oxford University Press, Oxford.

Margolin, A.A., Nemenman, I., Basso, K., Wiggins, C. Stolovitzky, G., Favera, R.D. and Califano, A., 2006. ARACNE: an algorithm for the reconstruction of gene regulatory networks ina mammalian cellular context. BMC Bioinformatics, 7(Suppl 1) S7.

Schäfer, J. and Strimmer, K., 2005. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Statistical Applications in Genetics and Molecular Biology, 4(1) article 32.

Gama-Castro, S. et al. 2008. RegulonDB (Version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and textpresso navigation. Nucleic Acids Research, vol 36, D120-D124.

Soranzo, N., Bianconi, G. and Altafini, C., 2007. Comparing association network algorithms for reverse engineering of large-scale gene regulatory networks: synthetic versus real data. Bioinformatics, 23(13) 1640-1647.

Steele, E. and Tucker, A., 2008. Consensus and meta-analysis regulatory networks for combining multiple microarray gene expression datasets. Journal of Biomedical Informatics, 41(6), 914-926.

Wang, Y., Joshi, T., Zhang, X., Xu, D. and Chen, L. 2006. Inferring gene regulatory networks from multiple microarray datasets. Bioinformatics, 22(19), 2413-2420.

Characteristic imset: a simple algebraic representative of a Bayesian network structure

Milan Studený
Institute of Information Theory and Automation of the ASCR, Czech Republic
studeny@utia.cas.cz

Raymond Hemmecke
TU Munich, Germany
hemmecke@ma.tum.de

Silvia Lindner
University of Magdeburg, Germany
lindner@mail.math.uni-magdeburg.de

Abstract

First, we recall the basic idea of an algebraic and geometric approach to learning a Bayesian network (BN) structure proposed in (Studený, Vomlel and Hemmecke, 2010): to represent every BN structure by a certain uniquely determined vector. The original proposal was to use a so-called *standard imset* which is a vector having integers as components, as an algebraic representative of a BN structure. In this paper we propose an even simpler algebraic representative called the *characteristic imset*. It is 0-1-vector obtained from the standard imset by an affine transformation. This implies that every reasonable quality criterion is an affine function of the characteristic imset. The characteristic imset is much closer to the graphical description: we establish a simple relation to any chain graph without flags that defines the BN structure. In particular, we are interested in the relation to the *essential graph*, which is a classic graphical BN structure representative. In the end, we discuss two special cases in which the use of characteristic imsets particularly simplifies things: learning decomposable models and (undirected) forests.

1 Introduction

The score and search method for learning Bayesian network (BN) structure from data consists in maximizing a *quality criterion* $\mathcal{Q}$, also named a *scoring criterion* or simply a *score* by other authors. It is a real function of the (acyclic directed) graph G and the observed database D. The value $\mathcal{Q}(G, D)$ measures how well the BN structure defined by G fits the database D.

Two important technical requirements on the criterion $\mathcal{Q}$ emerged in the literature in connection with computational methods dealing with this maximization task: $\mathcal{Q}$ should be *score equivalent* (Bouckaert, 1995) and (additively) *decomposable* (Chickering, 2002).

Another important question is how to represent the BN structure in the memory of a computer. It could be the case that different acyclic directed graphs are Markov *equivalent*, i.e., they define the same BN structure. A classic graphical characterization of equivalent graphs (Verma and Pearl, 1991) states that they are equivalent iff they have the same *adjacencies* and *immoralities*, which are special induced subgraphs. Representing a BN structure by any of the acyclic directed graphs defining it leads to a non-unique description causing later identification problems. Thus, researchers calling for methodological simplification proposed to use a unique representative for each individ-

Pp. 257–265 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

ual BN structure. The classic unique graphical representative is the *essential graph* (Andersson, Madigan and Perlman, 1997).

The idea of an algebraic approach, introduced in Section § 8.4 of (Studený, 2005), is to use an algebraic representative, called the *standard imset*. It is a vector whose components are integers indexed by subsets of the set of variables (= nodes) N. Moreover, it is a unique BN structure representative and the memory demands for its computer representation are polynomial in $|N|$. The most important point, however, is: Every score equivalent and decomposable criterion $\mathcal{Q}$ is an affine function (= linear function plus a constant) of the standard imset. Specifically, given an acyclic directed graph G (over N) and a database D, we have

$$\mathcal{Q}(G, D) \,=\, s_D^{\mathcal{Q}} - \langle t_D^{\mathcal{Q}}, u_G \rangle \,, \qquad (1)$$

where $s_D^{\mathcal{Q}}$ is a constant depending on the database and where $\langle t_D^{\mathcal{Q}}, u_G \rangle$ is the scalar product of a vector depending on the database, called the *data vector* (relative to $\mathcal{Q}$), and of the standard imset u_G (for G). Note that there is a polynomial-time algorithm (in $|N|$) for the reconstruction of the essential graph from the standard imset (Studený and Vomlel, 2009).

The geometric view was introduced in the paper (Studený, Vomlel and Hemmecke, 2010), where it was shown that the set of standard imset (over a fixed set of variables N) is the set of vertices (= extreme points) of a certain polytope. In particular, the maximization of $\mathcal{Q}$ over acyclic directed graphs can be re-formulated as a classic *linear programming problem*, that is, optimizing a linear function over a polyhedron.[1]

In this paper, we propose an alternative algebraic representative of a BN structure, called the *characteristic imset*. It is a vector obtained from the standard imset by a one-to-one affine transformation that maps lattice points to lattice points (in both directions). Thus, every score equivalent and decomposable criterion is an affine function of the characteristic imset and the set of characteristic imsets is the set of vertices of a polytope. The characteristic imset has only zeros and ones as its components. Moreover, it is very close to the graphical description: some of its components with value one correspond to adjacencies. Immoralities can also be recognized in the graph(s) on the basis of the characteristic imset. We establish a simple relation of the characteristic imset to any chain graph (without flags) defining the BN structure. In particular, this makes it possible to get immediately the characteristic imset on the basis of the essential graph. We also consider the converse task of reconstructing the essential graph from the characteristic imset.

If we restrict ourselves to decomposable models (= BN structures defined by acyclic directed graphs without immoralities), then the characteristic imset has a quite simple form. The situation is particularly transparent in the case of (undirected) forests: the edges of the forest are in one-to-one correspondence with 1's in the characteristic imset. The polytope spanned by these special characteristic imsets has already been studied in matroid theory (Schrijver, 2003). Consequently, we can easily learn (undirected) tree structures, which give an elegant geometric interpretation to a classic heuristic procedure proposed by Chow and Liu (1968).

The structure of this paper is as follows. In Section 2 we recall some of the definitions and relevant results. In Section 3 we introduce the characteristic imset and derive the above mentioned observations on it. Section 4 is devoted to the reconstruction of the essential graph from the characteristic imset. Section 5 briefly outlines our results about learning undirected forests from (Hemmecke et al., 2010). In Conclusions we discuss further perspectives.

2 Basic concepts

2.1 Graphical concepts

Graphs considered in this paper have a finite non-empty set of nodes N and two types of edges: directed edges, called *arrows*, denoted like $i \to j$ or $j \leftarrow i$, and undirected edges. No multiple edges are allowed between two nodes. If there is an edge between nodes i and j, we say they are *adjacent*.

[1] Note that a polytope is simply a bounded polyhedron.

Given a graph G over N and a non-empty set of nodes $A \subseteq N$, the *induced subgraph* of G for A has just those edges in G having both end-nodes in A. An *immorality* in G is an induced subgraph (of G) for three nodes $\{a, b, c\}$ in which $a \to c \leftarrow b$ and a and b are not adjacent. A *flag* is another induced subgraph for $\{a, b, c\}$ in which $a \to b$, b and c are adjacent by an undirected edge and a and c are not adjacent.

A set of nodes $K \subseteq N$ is *complete* in G if every pair of distinct nodes in K is adjacent by an undirected edge. A maximal complete set is called a *clique*. A set $C \subseteq N$ is *connected* if every pair of distinct nodes in C is connected via an undirected path. Maximally connected sets are called *components*.

A graph is *directed* if it has no undirected edges. A directed graph G over N is called *acyclic* if it has no directed cycle, that is, a sequence of nodes $a_1, \ldots, a_n$, $n \geq 3$ with $a_i \to a_{i+1}$ for $i = 1, \ldots n$, under the convention $a_{n+1} \equiv a_1$. An equivalent definition is the existence of an ordering $b_1, \ldots, b_m$, $m \geq 1$, of all nodes in N which is consistent with the direction of arrows, that is, $b_i \to b_j$ in G implies $i < j$.

A graph is *undirected* if it has no arrow. An undirected graph is called *chordal*, or *decomposable*, if every (undirected) cycle of length at least 4 has a chord, that is, an edge connecting two non-consecutive nodes in the cycle. There is a number of equivalent definitions for a decomposable graph (Lauritzen, 1996); one of them says that it is an undirected graph which can be acyclically directed without creating an immorality. A special case of a chordal graph is a *forest*, which is an undirected graph without undirected cycles. A forest over N in which N is connected is called a *(spanning) tree*.

A *chain graph* is a graph G (allowing both directed and undirected edges) whose components can be ordered into a chain, which is a sequence $C_1, \ldots, C_m$, $m \geq 1$ such that if $a \to b$ in G then $a \in C_i$ and $b \in C_j$ with $i < j$. An equivalent definition is: It is a graph without semi-directed cycles. Of course, every acyclic directed graph and every undirected graph is a special case of a chain graph (without flags).

Given a connected set C in a chain graph G, the set of *parents* of C is

$$pa_G(C) \equiv \{\, a \in N;\ a \to b \text{ in } G \text{ for some } b \in C \,\}.$$

Clearly, in a chain graph, $pa_G(C)$ is disjoint with (a connected set) C.

2.2 Bayesian network structures

Let N be a finite set of *variables*; to avoid the trivial case assume $|N| \geq 2$. For each $i \in N$ consider a finite individual *sample space* X_i (of possible values); to avoid technical problems assume $|\mathsf{X}_i| \geq 2$, for each $i \in N$. A *Bayesian network* can be introduced as a pair (G, P), where G is an acyclic directed graph having N as the set of its nodes and P a probability distribution on the *joint sample space* $\prod_{i \in N} \mathsf{X}_i$ that recursively factorizes according to G. Note that a factorization of P is equivalent to the condition that P is *Markovian* with respect to G meaning that it satisfies conditional independence restrictions determined by the respective separation criterion (Lauritzen, 1996).

BN structure (= Bayesian network structure) defined by an acyclic directed graph G is formally the class of probability distributions (on a fixed joint sample space) being Markovian with respect to G. Different graphs over N can be *Markov equivalent*, which means they define the same BN structure. The classic graphical characterization of (Markov) equivalent graphs is as follows (Verma and Pearl, 1991): they are equivalent if the have the same underlying undirected graph (= adjacencies) and the same immoralities. Of course, a BN structure can be described by any acyclic directed graph defining it, but there are other representatives (see below).

A complete *database* D of length $\ell \geq 1$ is a sequence $x_1, \ldots, x_\ell$ of elements of the joint sample space. By *learning BN structure* (from data) is meant to determine the BN structure based on an observed database D. A *quality criterion* is a real function $\mathcal{Q}$ of two variables: of an acyclic directed graph G and of a database D. The value $\mathcal{Q}(G, D)$ evaluates quantitatively how good the BN structure defined by G is to

explain the occurrence of the database D. However, we will not repeat the formal definition of the relevant concept of *statistical consistency* of $\mathcal{Q}$; see (Neapolitan, 2004). Since the aim is to learn a BN structure, a natural requirement is $\mathcal{Q}$ to be *score equivalent*, i.e., for fixed D, we have

$$\mathcal{Q}(G, D) = \mathcal{Q}(H, D),$$

for any pair of Markov equivalent acyclic directed graphs G and H over N.

An additively *decomposable* criterion (Chickering, 2002) is a criterion which can be written as follows:

$$\mathcal{Q}(G, D) = \sum_{i \in N} q_{i|pa_G(i)}(D_{\{i\} \cup pa_G(i)}),$$

where D_A for $\emptyset \neq A \subseteq N$ is the projection of the database D to $\prod_{i \in A} \mathsf{X}_i$ and $q_{i|B}$ for $i \in N$, $B \subseteq N \setminus \{i\}$ are real functions.

Statistical scoring methods are typically based on the likelihood function. For example, evaluating each BN structure by a *maximized log-likelihood* (MLL) leads to a score equivalent and additively decomposable criterion. However, this criterion is not statistically consistent in the sense of (Neapolitan, 2004), because it does not take the complexity of statistical models into consideration. Therefore, subtracting a penalty term evaluating the dimension of the statistical model and the length of the database may solve the problem. A standard example of such a criterion which is statistically consistent, score equivalent and decomposable is Schwarz's *Bayesian information criterion* (BIC) (1978).

2.3 Essential graph

The *essential graph* G^* of an equivalence class $\mathcal{G}$ of acyclic directed graphs over N is defined as follows:

- $a \to b$ in G^* if $a \to b$ in every G from $\mathcal{G}$,
- a and b are adjacent by an undirected edge in G^* if there are graphs G_1 and G_2 in $\mathcal{G}$ such that $a \to b$ in G_1 and $a \leftarrow b$ in G_2.

The first graphical characterization of essential graphs was provided by Andersson, Madigan and Perlman (1997).
It follows from this characterization that every essential graph is a chain graph without flags.

Actually, chain graphs without flags can serve as convenient graphical representatives of BN structures. As explained in Section 2.3 of (Studený, Roverato and Štěpánová, 2009), every chain graph defines a class of Markovian distributions, a statistical model, through the respective (generalized) separation criterion. As in case of acyclic directed graphs, they are called *Markov equivalent* if they define the same statistical model. Lemma 3 in (Studený, 2004) states that a chain graph H without flags is equivalent to an acyclic directed graph if the induced subgraphs for its components are chordal (undirected) graphs. Moreover, we can extend the graphical characterization of equivalence: two chain graphs without flags are Markov equivalent iff they have the same adjacencies and immoralities; see Lemma 2 in (Studený, 2004).

In this paper, we exploit the following characterization of essential graphs: Given an acyclic directed graph G, let $\mathcal{G}$ be the equivalence class of acyclic directed graphs containing G and $\mathcal{H}$ the (wider) equivalence class of chain graphs without flags containing G. The class $\mathcal{H}$ can be naturally (partially) ordered: if $H_1, H_2 \in \mathcal{H}$ and $a \to b$ in H_1 implies $a \to b$ in H_2 we call H_1 to be *larger* than H_2. With this partial ordering, the essential graph G^* (of $\mathcal{G}$) is just the largest graph in $\mathcal{H}$; see Corollary 4 in (Studený, 2004).

Moreover, there is a graphical procedure for getting G^* on the basis of any G in $\mathcal{G}$. It is based on a special graphical operation. Let H be a chain graph without flags. Consider two of its components, U called the *upper component* and L called the *lower component*. Provided the following two conditions hold:

- $pa_H(L) \cap U \neq \emptyset$ is a complete set in H,
- $pa_H(L) \setminus U = pa_H(U)$,

we say that the components can be *legally merged*. The result of merging is a graph obtained from H by replacing the arrows directed

from U to L into undirected edges. By Corollary 26 in (Studený, Roverato and Štěpánová, 2009), the resulting graph is also a chain graph without flags equivalent to H. Moreover, Corollary 28 in (2009) says: If G and H are equivalent chain graphs without flags and H is larger than G, then there exists a sequence of legal merging operations which successively transforms G into H. Of course, this is applicable to an acyclic directed graph G and the essential graph G^* in place of H.

2.4 Algebraic approach

In this paper, we consider vectors whose components are ascribed to (= indexed by) subsets of the set of variables N. Let $\mathcal{P}(N) \equiv \{A;\ A \subseteq N\}$ denote the power set of N. Every element of $\mathbb{R}^{|\mathcal{P}(\mathcal{N})|}$ can be written as a (real) combination of basic imsets vectors $\delta_A \in \{0,1\}^{|\mathcal{P}(\mathcal{N})|}$, $A \subseteq N$, where $\delta_A(A) = 1$ and $\delta_A(B) = 0$ for $A \neq B \subseteq N$.

Given an acyclic directed graph G over N, the *standard imset* for G in $\mathbb{R}^{|\mathcal{P}(\mathcal{N})|}$ is defined by the formula

$$u_G = \delta_N - \delta_\emptyset + \sum_{i \in N} \left\{ \delta_{pa_G(i)} - \delta_{\{i\} \cup pa_G(i)} \right\}, \quad (2)$$

where the basic vectors can cancel each other. An important fact is that two acyclic directed graphs G and H over N are Markov equivalent iff $u_G = u_H$; see Corollary 7.1 in (Studený, 2005). The crucial fact, however, is: Every score equivalent and decomposable criterion $\mathcal{Q}$ has the form (1), where $s_D^{\mathcal{Q}} \in \mathbb{R}$ and $t_D^{\mathcal{Q}} \in \mathbb{R}^{|\mathcal{P}(\mathcal{N})|}$ only depend on the data (and $\mathcal{Q}$); see Lemmas 8.3 and 8.7 in (2005). Moreover, (the constant $s_D^{\mathcal{Q}}$ and) the *data vector* $t_D^{\mathcal{Q}}$ is uniquely determined under additional standardization conditions $t_D^{\mathcal{Q}}(A) = 0$ for $A \subseteq N$ with $|A| \leq 1$.

For example, the standardized data vector for the MLL criterion can be computed as follows; see Proposition 8.4 in (2005). Let $\hat{P}$ denote the empirical measure on $\prod_{i \in N} \mathsf{X}_i$ computed from D and $\hat{P}_A$ its marginal for $A \subseteq N$. The *multiinformation* of $\hat{P}_A$ (for $A \neq \emptyset$) is its relative entropy $H(\hat{P}_A | \prod_{i \in A} \hat{P}_{\{i\}})$ with respect to the product of its own one-dimensional marginals. Then $t_D^{\mathsf{MLL}}(A) = \ell \cdot H(\hat{P}_A | \prod_{i \in A} \hat{P}_{\{i\}})$, where ℓ is the length of the database D. A formula for the data vector relative to the BIC criterion can be found in Section 8.4.2 of (Studený, 2005).

3 Characteristic imset

The characteristic imset is formally an element of $\mathbb{Z}^{|\mathcal{P}_*(\mathcal{N})|}$, where $\mathcal{P}_*(N) \equiv \{A \subseteq N;\ |A| \geq 2\}$ is the class of sets of cardinality at least 2.

Definition 1. Given an acyclic directed graph G over N, the *characteristic imset* for G is given by the formula

$$c_G(A) \ = \ 1 - \sum_{B, A \subseteq B \subseteq N} u_G(B)\,, \quad (3)$$

for $A \subseteq N$, $|A| \geq 2$.

Clearly, the characteristic imset is obtained from the standard one by an affine transformation of $\mathbb{R}^{|\mathcal{P}(\mathcal{N})|}$ to $\mathbb{R}^{|\mathcal{P}_*(\mathcal{N})|}$ (we only add and subtract entries of u_G). This mapping is invertible: We can compute back the standard imset by the formula

$$u_G(B) = \sum_{A, B \subseteq A \subseteq N} (-1)^{|A \setminus B|} \cdot (1 - c_G(A)) \quad (4)$$

for $B \subseteq N$, $|B| \geq 2$. The remaining values of u_G can then be determined by the formulas $\sum_{S \subseteq N} u_G(S) = 0$ and $\sum_{S, i \in S \subseteq N} u_G(S) = 0$ for $i \in N$. Since the transformation is one-to-one, two acyclic directed graphs G and H are equivalent iff $c_G = c_H$ (cf. Section 2.4). Thus, the characteristic imset is also a unique BN structure representative.

The basic observation is as follows; see also Theorem 3.2 in (Hemmecke et al., 2010):

Theorem 1. *For any acyclic directed graph G over N we have $c_G(A) \in \{0,1\}$ for any $A \subseteq N$, $|A| \geq 2$. Moreover, $c_G(A) = 1$ iff there exists $i \in A$ with $A \setminus \{i\} \subseteq pa_G(i)$.*

Proof. First, we substitute (2) into (3) and get for fixed $A \subseteq N$, $|A| \geq 2$:

$$\begin{aligned} c_G(A) \ &= \ - \sum_{i \in N, A \subseteq pa_G(i)} 1 + \sum_{i \in N, A \subseteq \{i\} \cup pa_G(i)} 1 \\ &= \sum_{i \in N, A \subseteq \{i\} \cup pa_G(i)\ \&\ i \in A} 1 = \sum_{i \in A, A \setminus \{i\} \subseteq pa_G(i)} 1\,. \end{aligned}$$

Assume for a contradiction there exist distinct $i, j \in A$ with $A \setminus \{i\} \subseteq pa_G(i)$ and $A \setminus \{j\} \subseteq pa_G(j)$. Then, however, both $j \to i$ and $i \to j$ are in G contradicting its acyclicity. In particular, $c_G(A) \in \{0, 1\}$. □

The consequence is the characterization of adjacencies and immoralities in terms of the characteristic imset.

Corollary 1. *Let G be an acyclic directed graph over N and a, b (and c) are distinct nodes. Then*

(i) *a and b are adjacent in G iff $c_G(\{a, b\}) = 1$.*

(ii) *$a \to c \leftarrow b$ is an immorality in G iff $c_G(\{a, b, c\}) = 1$ and $c_G(\{a, b\}) = 0$. The latter two conditions imply $c_G(\{a, c\}) = 1$ and $c_G(\{b, c\}) = 1$.*

Proof. Part (i) directly follows from Theorem 1: $c_G(\{a, b\}) = 1$ iff either $b \in pa_G(a)$ or $a \in pa_G(b)$. The necessity of the condition in (ii) also follows from Theorem 1. Conversely, if $c_G(\{a, b, c\}) = 1$, three options may occur: $\{b, c\} \subseteq pa_G(a)$, $\{a, c\} \subseteq pa_G(b)$ and $\{a, b\} \subseteq pa_G(c)$. But $c_G(\{a, b\}) = 0$ means by (i) that a and b are not adjacent in G, which excludes the first two options and implies that $a \to c \leftarrow b$ is an immorality in G. □

Now we show that any reasonable quality criteria is an affine function of the characteristic imset.

Definition 2. Given a score equivalent, additively decomposable criterion $\mathcal{Q}$ and a database D, let $t_D^{\mathcal{Q}}$ denote the standardized data vector relative to $\mathcal{Q}$. Introduce the *revised data vector* (relative to $\mathcal{Q}$) as an element of $\mathbb{R}^{|\mathcal{P}_*(\mathcal{N})|}$:

$$r_D^{\mathcal{Q}}(A) = \sum_{B, B \subseteq A, |B| \geq 2} (-1)^{|A \setminus B|} \cdot t_D^{\mathcal{Q}}(B) \qquad (5)$$

for $A \subseteq N$, $|A| \geq 2$.

Lemma 1. *Every score equivalent and additively decomposable criterion $\mathcal{Q}$ has the form*

$$\mathcal{Q}(G, D) \,=\, \mathcal{Q}(G^{\emptyset}, D) + \langle r_D^{\mathcal{Q}}, c_G \rangle \,, \qquad (6)$$

where $G^{\emptyset}$ is the graph over N without edges.

Proof. We substitute (4) into (1):

$$\mathcal{Q}(G, D) = s_D^{\mathcal{Q}} - \sum_{B \subseteq N, |B| \geq 2} t_D^{\mathcal{Q}}(B) \cdot \overbrace{\sum_{A, B \subseteq A} (-1)^{|A \setminus B|} \cdot (1 - c_G(A))}^{u_G(B)} \,.$$

Now, change the order of summation in the sum:

$$\sum_{A \subseteq N, |A| \geq 2} (1 - c_G(A)) \cdot \underbrace{\sum_{B \subseteq A, |B| \geq 2} (-1)^{|A \setminus B|} \cdot t_D^{\mathcal{Q}}(B)}_{r_D^{\mathcal{Q}}(A)} \,.$$

Thus, we get by (5):

$$\begin{aligned} \mathcal{Q}(G, D) &= s_D^{\mathcal{Q}} - \sum_{A \subseteq N, |A| \geq 2} (1 - c_G(A)) \cdot r_D^{\mathcal{Q}}(A) \\ &= \textit{constant} + \sum_{A \subseteq N, |A| \geq 2} c_G(A) \cdot r_D^{\mathcal{Q}}(A) \,. \end{aligned}$$

The observation that the characteristic imset for the empty graph $G^{\emptyset}$ is identically zero implies that the *constant* above is simply $\mathcal{Q}(G^{\emptyset}, D)$. □

Finally, we establish the relation of the characteristic imset to any chain graph without flags defining the BN structure.

Theorem 2. *Let H be a chain graph without flags equivalent to an acyclic directed graph G. For any $A \subseteq N$, $|A| \geq 2$ one has $c_G(A) = 1$ iff*

$$\exists\, \emptyset \neq K \subseteq A \textit{ complete in } H, \textit{ with } A \setminus K \subseteq pa_H(K). \quad (7)$$

Proof. In an acyclic directed graph G, the only non-empty complete sets are singletons. Thus, by Theorem 1, $c_G(A) = 1$ iff (7) holds with G (in place of H).

The next step is to observe that if $\tilde{H}$ is obtained from a chain graph H without flags by legal merging of components (see Section 2.3), then for any $A \subseteq N$, $|A| \geq 2$, (7) holds with H iff it holds with $\tilde{H}$. To verify this observe that any set A satisfying (7) has a uniquely determined component C with $K \subseteq C$ in H. Moreover, $pa_H(K) = pa_H(C)$, since H has no flags. The validity of (7) then depends on the induced subgraph of H for $C \cup pa_H(C)$. However, if $\tilde{H}$ is obtained from H by legal component merging, then most of these induced subgraphs are kept and the only change concerns the merged components U and L. We leave the reader to evidence that this change satisfies condition (7) in both directions.

Finally, we use the result mentioned in Section 2.3 which implies the existence of sequences of legal merging operations transforming G into G^* and H into G^*. In particular, for $A \subseteq N$, $|A| \geq 2$, (7) with G is equivalent to (7) with G^*, and this is equivalent to (7) with H. □

Of course, Theorem 2 applied to the essential graph G^* in place of H gives a direct method for obtaining the characteristic imset from the essential graph.

4 Back to the essential graph

Corollary 1 allows us to reconstruct the essential graph from the characteristic imset. Indeed, conditions (i) and (ii) determine both the adjacencies and immoralities (in every acyclic directed graph G defining the corresponding BN structure). Thus, we can directly get the *pattern* (of G) being the underlying undirected graph in which only the edges belonging to an immorality are directed.

This graph neither has to be the essential graph nor a chain graph. However, there is a simple (polynomial-time) procedure for transforming the pattern into the corresponding essential graph G^*. It consists of an (repeated) application of three orientation rules. Specifically, Theorem 3 in (Meek, 1995) states that the exhaustive application of rules from Figure 1 to the pattern of an acyclic directed graph G results in the essential graph (of the equivalence class containing G).

5 Learning undirected forests

Decomposable models (Lauritzen, 1996) can be viewed as BN structures whose essential graphs are (chordal) undirected graphs.

Corollary 2. *Let H be a chordal undirected graph over N. Then the corresponding characteristic imset c_H is specified as follows: $c_H(A) = 1$ iff A is a complete set in H.*

Proof. Consider the equivalence class $\mathcal{G}$ of acyclic directed graphs equivalent to H and apply Theorem 2. Since H has no arrow, (7) is equivalent to the above requirement. □

A special case of a chordal graph is an undirected forest. The only complete sets of cardinality at least 2 in it are its edges:

Corollary 3. *Let H be an undirected forest. Then the corresponding characteristic imset c_H vanishes for sets of cardinality 3 and more, and for distinct $a, b \in N$ we have $c_H(\{a, b\}) = 1$ iff a and b are adjacent in H.*

In particular, the characteristic imset for a forest can be identified with a vector of polynomial length $\binom{|N|}{2}$, which simplifies many things. For example, if maximizing a quality criterion $\mathcal{Q}$ over (undirected) forests is of interest, then, by Lemma 1, the function $H \mapsto c_H \in \mathbb{Z}^{|\mathcal{P}_*(\mathcal{N})|} \mapsto \langle r_D^{\mathcal{Q}}, c_H \rangle = \sum_{A \text{ edge in } H} r_D^{\mathcal{Q}}(A)$ should be maximized, that is, $H \mapsto \sum_{A \text{ edge in } H} t_D^{\mathcal{Q}}(A)$ by (5).

In particular, in case of the MLL criterion this means maximizing the sum of weights $\sum_{\{a,b\} \text{ edge}} w_{\{a,b\}}$, where $w_{\{a,b\}} = H(\hat{P}_{\{a,b\}} | \hat{P}_{\{a\}} \times \hat{P}_{\{b\}})$ is the (empirical) *mutual information* between a and b; see Section 2.4.

The polytope spanned by (restricted) characteristic imsets for forests has already been studied in matroid theory (Schrijver, 2003). It appears to be quite nice from an algorithmic point of view – for details see (Hemmecke et al., 2010). One important observation is the existence of a simple polynomial-time procedure based on the *greedy algorithm* for finding maximum-weight forest, where forests are weighted by the sums of weights of their edges.

This gives an elegant geometric interpretation to a classic (heuristic) procedure for approximating probability distributions with trees proposed by Chow and Liu (1968). Taking into account what was said above, it can be interpreted as the maximization of the MLL criterion over trees (= connected forests) using the greedy technique.

Conclusions

Our geometric interpretation of the classic learning procedure (for trees) may lead to useful generalizations. First, the application of the greedy algorithm is not limited to the MLL criterion and can be applied to maximize other

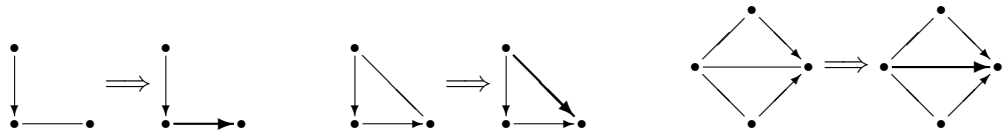


Figure 1: Orientation rules for getting the essential graph.

reasonable criteria like the BIC criterion. Second, we are not limited to trees and can apply the method to learning undirected forests, actually, to learning sub-forests of a prescribed undirected graph. Future research topics could be whether characteristic imsets can be applied to learning decomposable models, for example, with limited cardinality of cliques.

There are some related open questions. It follows from Section 4 that the components of the characteristic imset for sets of cardinalities 2 and 3 determine the remaining components. However, is there any direct method for determining them? Another question is whether Meek's (1995) orientation rules can be avoided in the reconstruction of the essential graph on the basis of the characteristic imset. We hope that a modification of the procedure from (Studený and Vomlel, 2009) leads to such an algorithm.

Acknowledgments

This research has been supported by the grants GAČR n. 201/08/0539 and MŠMT n. 1M0572.

References

Steen A. Andersson, David Madigan and Michael D. Perlman. 1997. A characterization of Markov equivalence classes for acyclic digraphs. *Annals of Statistics*, 25(2):505–541.

Remco R. Bouckaert. 1995. Bayesian belief networks: from construction to evidence. PhD thesis, University of Utrecht.

David M. Chickering. 2002. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

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

Raymond Hemmecke, Silvia Lindner, Milan Studený, and Jiří Vomlel. 2010. Characteristic imsets for learning Bayesian network structures. In preparation.

Steffen L. Lauritzen. 1996. *Graphical Models*, Clarendon Press.

Chris Meek. 1995. Causal inference and causal explanation with background knowledge. In *11th Conference on Uncertainty in Artificial Intelligence*, pages 403–410.

Richard E. Neapolitan. 2004. *Learning Bayesian Networks*, Pearson Prentice Hall.

Alexander Schrijver. 2003. *Combinatorial Optimization - Polyhedra and Efficiency, volume B*, Springer Verlag.

Gideon E. Schwarz. 1978. Estimation of the dimension of a model. *Annals of Statistics*, 6:461–464.

Milan Studený. 2004. Characterization of essential graphs by means to the operation of legal merging of components. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12:43–62.

Milan Studený. 2005. *Probabilistic Conditional Independence Structures*, Springer Verlag.

Milan Studený and Jiří Vomlel. 2009. A reconstruction algorithm for the essential graph. *International Journal of Approximate Reasoning*, 50:385–413.

Milan Studený, Alberto Roverato and Šárka Štěpánová. 2009. Two operations of merging and splitting components in a chain graph. *Kybernetika*, 45(2):208–248.

Milan Studený, Jiří Vomlel and Raymond Hemmecke. 2010. A geometric view on learning Bayesian network structures. *International Journal of Approximate Reasoning*, 51(5):578–586.

Thomas Verma and Judea Pearl. 1991. Equivalence and synthesis of causal models. In *6th Conference on Uncertainty in Artificial Intelligence*, pages 220–227.

Multilabel Classification of Drug-like Molecules via Max-margin Conditional Random Fields

Hongyu Su
University of Helsinki, Finland
hongyu.su@cs.helsinki.fi

Markus Heinonen
University of Helsinki, Finland
markus.heinonen@cs.helsinki.fi

Juho Rousu
University of Helsinki, Finland
juho.rousu@cs.helsinki.fi

Abstract

We present a multilabel learning approach for molecular classification, an important task in drug discovery. We use a conditional random field to model the dependencies between drug targets and discriminative training to separate correct multilabels from incorrect ones with a large margin. Efficient training of the model is ensured by conditional gradient optimization on the marginal dual polytope, using loopy belief propagation to find the steepest feasible ascent directions. In our experiments, the MMCRF method outperformed the support vector machine with state-of-the-art graph kernels on a dataset comprising of cancer inhibition potential of drug-like molecules against a large number cancer cell lines.

1 Introduction

Machine learning has become increasingly important in drug discovery, where viable molecular structures are searched or designed for therapeutic efficacy. In particular, the costly preclinical *in vitro* and *in vivo* testing of drug candidates can be focused to the most promising molecules, if accurate *in silico* models are available (Trotter et al., 2001).

Molecular classification has been tackled with a variety of methods, including inductive logic programming (King et al., 1996) and artificial neural networks (Bernazzani et al., 2006). During the last decade kernel methods (Ralaivola et al., 2005; Swamidass et al., 2005; Trotter et al., 2001; Ceroni et al., 2007) have emerged as a computationally effective way to handle the non-linear properties of chemicals. In numerous studies, SVM-based methods have obtained promising results (Byvatov et al., 2003; Trotter et al., 2001; Zernov et al., 2003). However, classification methods focusing on a single target variable are probably not optimally suited to drug screening applications where large number of target cell lines are to be handled.

Multilabel classification, where the objects can be classified into more than one category at a time, have received a significant attention in recent years both in hierarchical (Silla and Freitas, 2010) and Bayesian network settings (de Waal and van der Gaag, 2007; Rodriguez and Lozano, 2008). In this paper we propose, to our knowledge, the first application of multilabel learning to molecular classification. Our learning method belongs to the structured output prediction family (Taskar et al., 2003; Tsochantaridis et al., 2004; Rousu et al., 2006; Rousu et al., 2007); the drug targets (cancer cell lines) are organized in a Markov network, drug molecules are represented by kernels and max-margin training is used to learn the

Pp. 265–273 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

parameters. Loopy belief propagation over the Markov network is used both in learning the model and in extracting the predicted multilabel.

2 Multilabel learning with MMCRF

The model used in this paper is an instantiation of the Max-Margin Conditional Random Field (MMCRF) framework (Rousu et al., 2007) for associative Markov networks and can also seen as a sibling method to HM^3 (Rousu et al., 2006), which is designed for hierarchies. Here we give an overview of the method for transparency, the interested reader may check the details from the above references.

We consider data from a domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is a set and $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k$ is a Cartesian product of the sets $\mathcal{Y}_j = \{+1,-1\}, j = 1, \ldots, k$. A vector $\mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y}$ is called the *multilabel* and the components y_j are called the *microlabels*.

We assume that a training set $\{(x_i, \mathbf{y}_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}$ has been given. In addition, a pair $(x_i, \mathbf{y})$ where x_i is a training pattern and $\mathbf{y} \in \mathcal{Y}$ is arbitrary, is called a *pseudo-example*, to denote the fact that the pair may or may not be generated by the distribution generating the training examples. As the model class we use the exponential family

$$P(\mathbf{y}|x) \propto \prod_{e \in E} \exp\left(\mathbf{w}_e^T \varphi_e(x, \mathbf{y}_e)\right)$$

defined on the edges of a Markov network $G = (V, E)$, where node $j \in V$ corresponds to the j'th component of the multilabel and the edges $e = (j, j') \in E$ correspond to a microlabel dependency structure given as input. By $\mathbf{y}_e = (y_j, y_{j'})$ we denote the pair of microlabels of the edge $e = (j, j')$ and $\varphi(x, \mathbf{y}) = (\varphi_e(x, \mathbf{y}_e))_{e \in E}$ is a *joint feature map* for inputs and outputs. The joint feature map is given by the tensor product $\varphi(x, \mathbf{y}) = \phi(x) \otimes \psi(\mathbf{y})$ of input features $\phi(x)$ computed from the molecules (see Section 3) and output features $\psi(\mathbf{y}) = (\psi_{eu}(\mathbf{y}))$ corresponding to possible labelings $u \in \{+1, -1\}^2$ of the edges $e \in E$: $\psi_{eu}(\mathbf{y}) = [\![\mathbf{y}_e = u]\!]$. The tensor product then contains all pairs $\phi_r(x) \cdot \psi_{eu}(\mathbf{y})$ of input and output features. The benefit of the tensor product representation is that context (edge-labeling) sensitive weights can be learned for input features and no prior alignment of input and output features need to be assumed.

2.1 Max margin learning

To learn the parameters of the model we apply margin-based structured output prediction (c.f. (Taskar et al., 2003; Tsochantaridis et al., 2004)). The primal optimization problem takes the form

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \ & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \ & \mathbf{w}^T \Delta\varphi(x_i, \mathbf{y}) \geq \ell(\mathbf{y}_i, \mathbf{y}) - \xi_i, \\ & \text{for all } i \text{ and } \mathbf{y}, \end{aligned} \tag{1}$$

where $\Delta\varphi(x_i, \mathbf{y}) = \varphi(x_i, \mathbf{y}_i) - \varphi(x_i, \mathbf{y})$, $\ell(\mathbf{y}_i, \mathbf{y})$ is the loss of the pseudo-example, ξ_i is the slack and the parameter C controls the amount of regularization in the model. The corresponding Lagrangian dual problem takes the form:

$$\begin{aligned} \underset{\alpha \geq 0}{\text{maximize}} \ & \alpha^T \ell - \frac{1}{2} \alpha^T K \alpha \\ \text{s.t.} \ & \sum_{\mathbf{y}} \alpha(i, \mathbf{y}) \leq C, \forall i, \mathbf{y}, \end{aligned} \tag{2}$$

where $K = \left(\Delta\varphi(x_i, \mathbf{y})^T \Delta\varphi(x_j, \mathbf{y}')\right)$ is the *joint* kernel matrix for *pseudo-examples* $(x_i, \mathbf{y})$ and $\ell = (\ell(\mathbf{y}_i, \mathbf{y}))_{i,\mathbf{y}}$ encodes the loss for each $(x_i, \mathbf{y})$.

2.2 Marginal dual problem

The above optimization problems are challenging due to the exponential number of constraints or dual variables. A more manageable sized problem is obtained by considering the edge-marginals of dual variables

$$\mu(i, e, u) = \sum_{\mathbf{y} \in \mathcal{Y}} \psi_{eu}(\mathbf{y}) \alpha(i, y), \tag{3}$$

where $e \in E$ is an edge in the output network and $u \in \{+1, -1\}^2$ is a possible labeling for the edge. Using the marginal dual representation, we can state the dual problem (2) in equivalent form as (for details, see (Rousu et al., 2007)):

$$\max_{\mu \in \mathcal{M}^m} \ \mu^T \ell - \frac{1}{2} \mu^T K_E \mu, \tag{4}$$

where $\mathcal{M}$ denotes the marginal polytope[1] (c.f. (Wainwright et al., 2005)), the set of all combinations of marginal variables (3) of an example that have a counterpart in the dual feasible set in (2), $K_E = diag(K_e)_{e \in E}$ contains the joint kernel values pertaining to the edges, $\mu = (\mu(i, e, v))$ is the vector of marginal dual variables and $\ell = (\ell(i, e, v))$ is the vector of losses between edge-labelings, $\ell(i, e, v) = \ell(\mathbf{y}_{ie}, v)$. This problem is a quadratic programme with a number of variables *linear* in both the size of the output network and the number of training examples. Thus, there is an exponential reduction in the number of dual variables from the original dual (2).

2.3 Conditional gradient optimization

The marginal dual problem is solved by an iterative optimization algorithm where the marginal dual variables of each example in turn are optimized using conditional gradient algorithm whilst keeping the other training examples fixed. The conditional gradient step (Algorithm 1) iteratively finds the best feasible direction given the current subgradient $g_i = \ell_i - K_{i\cdot}\mu$ of the objective

$$\mu_i^* = \underset{\mathbf{v} \in \mathcal{M}}{\mathbf{argmax}}\ g_i^T \mathbf{v} \tag{5}$$

and uses exact line search to locate the optimal point in that direction.

The feasible ascent directions in (5) correspond to vertices[2] of the marginal dual polytope $\mathcal{M}$ which via (3) are images of the vertices of the original dual set and thus have one to one correspondence to the set of possible multilabels.

Consequently, for each solution μ_i^* of (5) there is a corresponding multilabel $\mathbf{y}^*$ which, comparing equations (5) and (1), can be seen as the pseudo-example $(x_i, \mathbf{y})$ that violates its margin maximally. Instead of (5) we can thus solve the multilabel with maximum gradient

$$\begin{aligned} \mathbf{y}^* &= \underset{\mathbf{y}}{\mathbf{argmax}}\ g_i^T \mu_i^*(\mathbf{y}) \\ &= \underset{\mathbf{y}}{\mathbf{argmax}} \sum_{e \in E} g(i, e, \mathbf{y}_e)\mu(i, e, \mathbf{y}_e) \end{aligned} \tag{6}$$

and return the corresponding vertex $\mu^* = \mu(\mathbf{y}^*)$ of the marginal dual polytope. The problem (6) is readily seen as an inference problem on the Markov network G: one must find the configuration $\mathbf{y}^*$ that maximizes the sum of the 'edge potentials' $g(i, e, \mathbf{y}_e)\mu(i, e, \mathbf{y}_e)$.

Inference on a general graph is known to be hard. However, for our purposes, an approximate solution suffices: within an iterative algorithm it does not pay to spend a lot of time looking for optimal ascent direction when a reasonable ascent direction can be found fast. Here we opt to use loopy belief propagation with early stopping: we only compute a few iterations (given by the user parameter $maxLBPiter$) of the inference before returning (row 3 in Algorithm 1).

In addition to being the workhorse for optimizing the classification model, loopy belief propagation is also used in the prediction phase to extract the model's prediction given the learned parameters.

Algorithm 1 Conditional gradient inference for single example.

Input: Initial dual variable vector μ_i, gradient g_i, a joint kernel block K_{ii} for the subspace
Output: New values for dual variables μ_i.

1: $iter = 0$;
2: **while** $iter < maxcgiter$ **do**
3: $\quad \mu_i^* = feasibleDir(\mathbf{g}_i, E, maxLBPiter)$;
4: $\quad \tau = lineSearch(\mu_i^*, \mu_i, K_{ii})$;
5: $\quad$ **if** $\tau \leq 0$ **then**
6: $\quad\quad$ **break**; % no progress, stop
7: $\quad$ **else**
8: $\quad\quad \mu_i = \mu_i + \tau(\mu^* - \mu_i)$; % new solution
9: $\quad\quad g_i = g_i - \tau K_{ii}(\mu^* - \mu_i)$; % new gradient
10: $\quad$ **end if**
11: $\quad iter = iter + 1$;
12: **end while**

[1] We use the same probabilistic interpretation of dual variables as (Taskar et al., 2003).

[2] In the presence of ties, there is a set of vertices with optimum score; ties can be broken arbitrarily.

3 Kernels for drug-like molecules

Prediction of bioactivity is typically based on the physico-chemical and geometric properties of the molecules. Kernels computed from the structured representation of molecules extend the scope of the traditional approaches by allowing complex derived features to be used while avoiding excessive computational cost (Ralaivola et al., 2005). In this section, we will review the main approaches to construct a graph kernel for classification of drug-like molecules.

3.1 Walk Kernel

The classic way to represent the structure of a molecule is to use an undirected labeled graph $G = (V, E)$, where vertices $V = \{v_1, v_2, \ldots, v_n\}$ corresponds to the atoms and edges $E = \{e_1, e_2, \ldots, e_m\}$ to the covalent bonds. Vertex labels correspond to atom types (e.g. "oxygen", "carbon", etc.), and edge labels correspond to bond types (e.g. "single", "double", "aromatic", etc.). The $n \times n$ adjacency matrix E of graph G is defined such that its (i, j)'th entry E_{ij} equals to one if and only if there is an edge between vertices v_i and v_j.

Walk kernels (Kashima et al., 2003; Gärtner, 2003) compute the sum of matching walks in a pair of graphs. The contribution of each matching walk is downscaled exponentially according to its length. A *walk* of length m in a graph G is denoted by $w = \{v_1, v_2, ..., v_m\}$ such that for $i = 1, 2, ..., m-1$ there exists an edge for each pair of vertices (v_i, v_{i+1}).

A direct product graph between two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is denoted by $G_\times(G_1, G_2)$. Vertices of a product graph $G_\times(G_1, G_2)$ are defined as

$$V_\times(G_1, G_2) = \{(v_1, v_2) \in V_1 \times V_2, \; label(v_1) = label(v_2)\},$$

and edges are defined as

$$E_\times(G_1, G_2) = \{((v_1, v_2), (u_1, u_2)) \in V_\times \times V_\times, \; (v_1, u_1) \in E_1 \wedge (v_2, u_2) \in E_2\}.$$

The walk kernel can be defined as

$$K_{wk}(G_1, G_2) = \sum_{i,j=1}^{|v_\times|} \left[\sum_{n=0}^{\infty} \lambda^n E_\times^n \right]_{ij},$$

where $v_\times$ is the vertex in product graph, λ^n is the positive downscaling factor that is strictly less than one and n is the length of walk.

Since longer walks are downscaled by λ^n, the contribution of longer walks are ofter negligible. Therefore, we consider finite-length walk kernel where only walks of length p are explicitly constructed:

$$K_{wk_p}(G_1, G_2) = \sum_{v_i \in V_\times} D_p(v_i),$$

where $D_p(v_i)$ is calculated in a dynamic programming fashion by

$$D_0(v_i) = 1,$$
$$D_n(v_i) = \sum_{v_i, v_j \in E_\times} D_{n-1}(v_j).$$

3.2 Weighted decomposition kernel

The weighted decomposition kernel is an extension of the substructure kernel by weighting identical parts in a pair of graphs based on contextual information (Ceroni et al., 2007).

A weighted decomposition kernel on a labeled graph G is based on a decomposition $D_r(G) = \{(s, z) : s \in V, z = \mathcal{N}_r(s)\}$, where $\mathcal{N}_r(s)$ is the neighborhood subgraph of radius r of vertex s. The s is called *selector*, and z is the subgraph around it called *contextor*. A kernel function between two graphs G_1 and G_2 is defined as

$$K_{wdk}(G_1, G_2) = \sum_{\substack{v,z \in D_r(G_1) \\ v',z' \in D_r(G_2)}} [\![v = v']\!] K_s(z, z'),$$

where $K_s(z, z')$ is the kernel function between a pair of contextors. The function $K_s(z, z')$ uses the subgraph histogram intersection kernel discarding subgraph structure information defined as

$$K_s(z, z') = \sum_{l \in L} K_r(z, z'),$$
$$K_r(z, z') = \sum_{j=1}^{m_l} \min\{p_l(j), p'_l(j)\},$$

where L is total number of attributes labeled on each vertex, m_l is the number of possible values of the l'th property, and $p_l(j)$, $p'_l(j)$ are the observed frequencies of value j for l'th attribute for subgraphs z and z', respectively.

3.3 Molecular fingerprints and the Tanimoto kernel

Molecular fingerprints are designed to encode a molecular structure into a fixed width binary bit vector that represents the presence or absence of substructures or fragments in the molecule. Molecular fingerprints are extensively used in chemical informatics.

There are two main types of fingerprints. *Hash fingerprints* enumerate all linear fragments of length n in a molecule. Parameter n is usually bounded from three to seven. A hash function assigns each fragment a hash value, which determines its position in descriptor space.

Another major fingerprint type is *substructure keys*, which is based on a pattern matching of a molecular structure to a set of pre-defined substructures. Each substructure becomes a key and has a fixed position in descriptor space. These substructures are considered to be independent functional units identified by domain experts as prior knowledge.

Once the molecules have been represented as fingerprints, the Tanimoto kernel (Ralaivola et al., 2005) is usually employed to measure the similarity between a pair of molecules. Given two molecular fingerprints fp_1 and fp_2, the Tanimoto kernel is defined as

$$K_{tk}(fp_1, fp_2) = \frac{N_{fp_1,fp_2}}{N_{fp_1} + N_{fp_2} - N_{fp_1,fp_2}},$$

where N_{fp_1} is the number of 1-bits in fingerprint fp_1, N_{fp_2} is the number of 1-bits in fingerprint fp_2, and N_{fp_1,fp_2} is the number of 1-bits in both of the fingerprints.

4 Experiments

4.1 NCI-Cancer dataset

In this paper we use the NCI-Cancer dataset obtained through PubChem Bioassay[3] (Wang et al., 2009) data repository. The dataset initiated by National Cancer Institute and National Institutes of Health (NCI/NIH) contains bioactivity information of large number of molecules against several human cancer cell lines in 9 different tissue types, including leukemia, melanoma and cancers of the lung, colon, brain, ovary, breast, prostate, and kidney. For each molecule tested against a certain cell line, the dataset provides the bioactivity outcome that we use as the classes (active, inactive).

However, molecular activity data are highly biased over the cell lines. Figure 1 shows the molecular activity distribution over all 59 cell lines. Most of the molecules are inactive in all cell lines, while a relatively large proportion of molecules are active against almost all cell lines, which can be taken as toxics. These molecules are less likely to be potential drug candidates than the ones in the middle part of the histogram.

In order to circumvent the skewness and to concentrate on the most interesting molecules, we adopted the preprocessing suggested in (Shivakumar and Krauthammer, 2009), and selected molecules that are active in more than 10 cell lines and inactive in more than 10 cell lines. As a result, 544 molecules remained and were employed in our experiments.

4.2 Experiment setup and measures of success

To circumvent the skewness of the multilabel distribution, we use the following stratified cross-validation scheme to compare the methods: we divide examples into pools by the number of cell lines each molecule is active in (c.f. Figure 1). Then, we divide each pool into five

[3] http://pubchem.ncbi.nlm.nih.gov

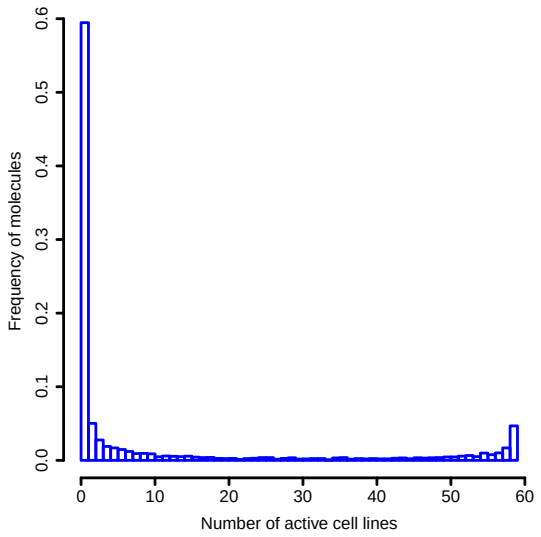


Figure 1: Skewness of the multilabel distribution.

folds and finally merge the corresponding pool-specific folds into five global folds.

To compare the performace of SVM and MMCRF in the multilabel prediction tasks we use microlabel accuracy and microlabel F1 score: we pool together individual microlabel predictions over all examples and all cell lines, and count accuracy and F1 from the pool.

4.3 Markov network generation for cancer cell lines

There are many ways one can build the Markov network in between the cell lines to be used as input to the MMCRF algorithm. For the dataset used in this paper, a large set of auxiliary data is available on the cancer cell lines from the NCI database[4]. Based on preliminary tests we opted to use RNA Radiation Array data. The basic approach is to construct from this data a correlation matrix between the pairs of cell lines and extract the Markov network from the matrix by favoring high-valued pairs. The following methods of network extraction were considered:

SPT. Maximum weight spanning tree. Take the minimum number of edges that make a connected network whilst maximizing the edge weights.

CTh. Correlation thresholding. Take all edges that exceed fixed threshold. This approach typically generates a general non-tree graph.

Rnd. Random graph. Draw edges uniformly at random.

Table 1: MMCRF Accuracy and F1 score with different Markov network extraction methods

	RNA Rad. array		
	SPT	***CTh***	***Rnd***
Accuracy	**67.6%**	65.1%	66.3%
F1 Score	**56.2%**	52.8%	53.5%

Table 2: Accuracies and microlabel F1 scores from different kernels in MMCRF and SVM.

Methods	***Accuracy***	***F1 score***
SVM + WK	64.6%	49.0%
SVM + WDK	63.9%	51.6%
SVM + Tanimoto	64.1%	52.7%
MMCRF + Tanimoto	**67.6%**	**56.2%**

In our experiments, the spanning tree approach on RNA radiation array data turned out to be best approach on average (Table 1). Suprisingly, correlation thresholding fails to meet the accuracy of random graph. However, the predictive performance of MMCRF surpasses SVM (c.f. Table 2) regardless of the network generation method.

4.4 Effect of molecule kernels

We conducted experiments to compare the effect of various kernels, as well as the performances of support vector machine (SVM) and MMCRF. We used the SVM implementation of the LibSVM software package written in C++[5]. We tested SVM with different margin C parameters, relative hard margin ($C = 100$) emerging

[4] http://discover.nci.nih.gov/cellminer/home.do

[5] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

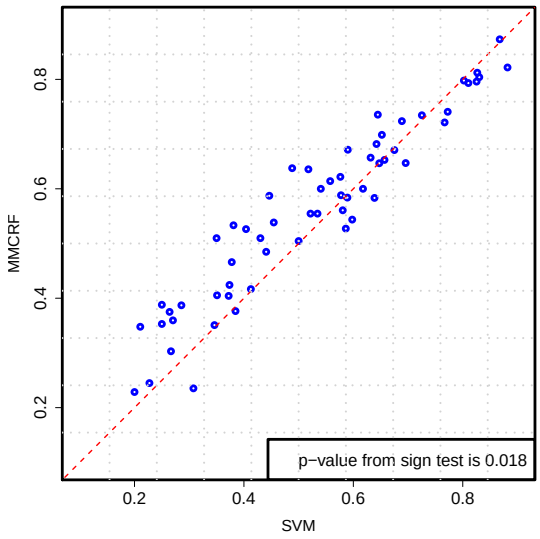


Figure 2: Microlabel F1 score of MMCRF against SVM classifier in each cell line.

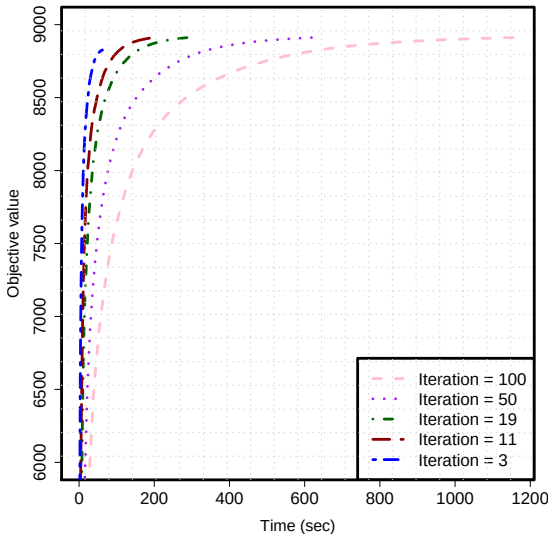


Figure 3: Effect of loopy belief propagation iteration.

as the value used in subsequent experiments. The same value was used for MMCRF classifier as well.

For the three kernel methods, walk kernel (WK) was constructed using parameters $\lambda = 0.1$ and $p = 6$ as recommended in (Gärtner, 2003). The Weighted decomposition kernel (WDK) used context radius $r = 3$ as in (Ceroni et al., 2007), and a single attribute (atom type) was sufficient to give the best performance. We also used hash fragments as molecular fingerprints generated by OpenBabel[6] (using default value $n = 6$ for linear structure length), which is a chemical toolbox available in public domain. All kernels were normalized.

In Table 2, we report overall accuracies and microlabel F1 scores using SVM with different kernel methods. The kernels achieve almost the same accuracy within SVM, while Tanimoto kernel is slightly better than others in microlabel F1 score. We further compared MMCRF and SVM classifiers with Tanimoto kernel. MMCRF turned out to outperform SVM in both overall accuracy and microlabel F1 score.

Figure 2 gives the F1 score in each cell line from MMCRF classifier against SVM classifier in the same experiment. Points above the diagonal line correspond to improvements in F1 scores by MMCRF classifier. MMCRF improves microlabel F1 scores of 39 out of 59 cell lines with sign test giving the p-value of 0.018. The statistics for accuracy were similar (data not shown).

[6]http://openbabel.org

4.5 Effect of loopy belief propagation

Finally, we tested different loopy belief propagation iteration parameters to see the their effects on convergence (Figure 3). The best loopy belief propagation iteration limit turned out to be $maxLBPiter = 11$, Smaller values were not sufficient for MMCRF to reach a global optimum, while larger values caused the convergence to need more time. The optimal value turned out to be close to the diameter of the Markov network (10 in this case), indicating that propagation of messages through the whole network is required for best performance.

5 Conclusions

We presented a multilabel classification approach to drug activity classification using the Max-Margin Conditional Random Field algorithm. In experiments against a large set of cancer lines the method significantly outperformed SVM.

Acknowledgements

This work was financially supported by Academy of Finland grant 118653 (ALGODAN) and in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.

References

L. Bernazzani, C. Duce, A. Micheli, V. Mollica, A. Sperduti, A. Starita, and M.R. Tine. 2006. Predicting physical-chemical properties of compounds from molecular structures by recursive neural networks. *J. Chem. Inf. Model.*, 46:2030–2042.

E. Byvatov, U. Fechner, J. Sadowski, and G. Schneider. 2003. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *J. Chem. Inf. Comput. Sci.*, 43:1882–1889.

A. Ceroni, F. Costa, and P. Frasconi. 2007. Classification of small molecules by two- and three-dimensional decomposition kernels. *Bioinformatics*, 23:2038–2045.

P. de Waal and L. van der Gaag. 2007. Inference and Learning in Multi-dimensional Bayesian Network Classifiers. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 501–511.

T. Gärtner. 2003. A survey of kernels for structured data. *SIGKDD Explor. Newsl.*, 5(1):49–58.

H. Kashima, K. Tsuda, and A. Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the* 20th *International Conference on Machine Learning (ICML)*, Washington, DC, United States.

R. King, S. Muggleton, A. Srinivasan, and M. Sternberg. 1996. Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *PNAS*, 93:438–442.

L. Ralaivola, S. Swamidass, H. Saigo, and P. Baldi. 2005. Graph kernels for chemical informatics. *Neural Networks*, 18:1093–1110.

JD Rodriguez and J.A. Lozano. 2008. Multi-objective learning of multi-dimensional bayesian classifiers. In *Eighth International Conference on Hybrid Intelligent Systems, 2008. HIS'08*, pages 501–506.

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. 2006. Kernel-Based Learning of Hierarchical Multilabel Classification Models. *The Journal of Machine Learning Research*, 7:1601–1626.

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. 2007. Efficient algorithms for max-margin structured classification. *Predicting Structured Data*, pages 105–129.

P. Shivakumar and M. Krauthammer. 2009. Structural similarity assessment for drug sensitivity prediction in cancer. *Bioinformatics*, 10:S17.

C.N. Silla and A.A. Freitas. 2010. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, pages 1–42.

S.J. Swamidass, J. Chen, J. Bruand, P. Phung, L. Ralaivola, and P. Baldi. 2005. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21:359–368.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *Neural Information Processing Systems 2003*.

M. Trotter, M. Buxton, and S. Holden. 2001. Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Comp. and Chem.*, 26:1–20.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y.n Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. 21th International Conference on Machine Learning*, pages 823–830.

M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. 2005. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717.

Y. Wang, E. Bolton, S. Dracheva, K. Karapetyan, B.A. Shoemaker, T.O. Suzek, J. Wang, J. Xiao, J. Zhang, and S.H. Bryant. 2009. An overview of the pubchem bioassay resource. *Nucleic Acids Research*, 38:D255–D266.

V. Zernov, K. Balakin, A. Ivaschenko, N. Savchuk, and I. Pletnev. 2003. Drug discovery using support vector machines. The case studies of drug-likeness, agrochemical-likeness, and enzyme inhibition predictions. *J. Chem. Inf. Comput. Sci.*, 43:2048–2056.

Reinforcing the Object-Oriented Aspect of Probabilistic Relational Models

Lionel Torti - Pierre-Henri Wuillemin - Christophe Gonzales
LIP6 - UPMC - France
firstname.lastname@lip6.fr

Abstract

Representing uncertainty in knowledge is a common issue in Artificial Intelligence. Bayesian Networks have been one of the main models used in this field of research. The simplicity of their specification is one of the reason for their success, both in industrial and in theoretical domains. The widespread use of Bayesian Networks brings new challenges in the design and use of large-scale systems, where this very simplicity causes a lack of expressiveness and scalability. To fill this gap, an increasing number of languages emerged as extensions of Bayesian Networks with many approaches: first-order logic, object-oriented, entity-relation, and so on. In this paper we focus on Probabilistic Relational Models, an object-oriented extension. However, Probabilistic Relational Models do not fully exploit the object-oriented paradigm, in particular they lack class inheritance. Using Object-Oriented Bayesian Networks as a basis, we propose to lightly extend PRMs framework resulting in stronger object-oriented aspects in probabilistic models.

Probabilistic graphical models (Koller and Friedman, 2009) are a general purpose framework for dealing with uncertainty. Their applications to many different domains has stimulated an uninterrupted process of creation of new frameworks based on probability theory. Bayesian Networks (Pearl, 1988) are among the most popular framework for uncertainty in AI.

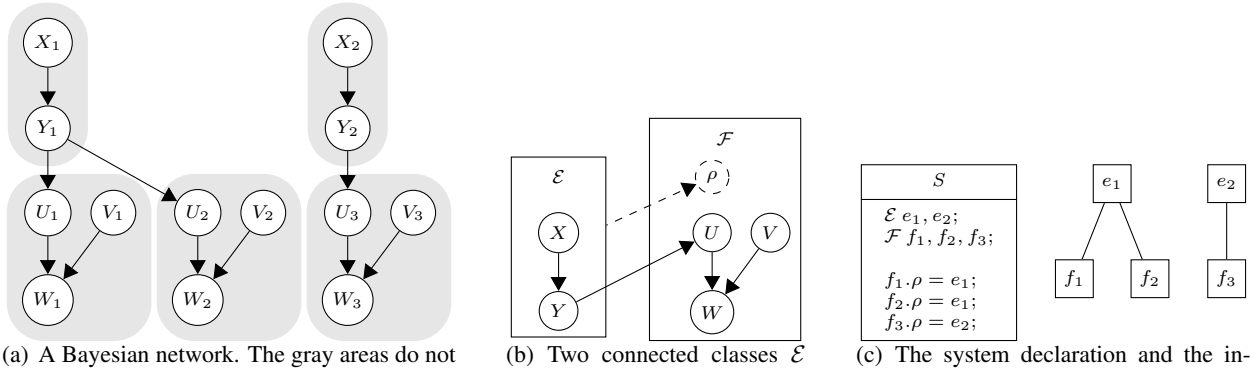
In recent years, the Statistical Learning community has actively proposed new probabilistic frameworks, closing the gap between first-order logic and probability theory (Getoor and Taskar, 2007). New models such as Object-Oriented Bayesian Networks (Koller and Pfeffer, 1997; Bangsø and Wuillemin, 2000a), Multiply-Sectioned Bayesian Networks (Yang, 2002), Probabilistic Relational Models (Getoor et al., 2007) and Multi-Entity Bayesian Networks (Laskey, 2008) have extended Bayesian Networks and widen their range of application.

In many situations, these new first-order logic-based networks can be efficiently learned from databases and used for answering probabilistic queries. However, there are situations like nuclear plant safety problems where the scarcity of data available prevents such learning. For such problems, oriented graphical models such as Probabilistic Relational Models (PRMs) are often more suitable than the aforementioned first-order models because they can often be modeled by interactions with experts of the domain.

PRMs have an object-oriented basis, but they lack fundamental mechanisms related to class inheritance. In software engineering, such object-oriented designs has proved very useful for creating complex software. In this paper, we illustrate why these mechanisms are necessary for practical design of large-scale systems and we show how light extensions can enforce strong object-oriented features into the PRMs framework. In addition, we propose a representation of PRMs with such mechanisms using parfactors, the state-of-the-art framework for first-order probabilistic inference (Poole, 2003). All the concepts we present here are implemented in our open source C++ framework called *aGrUM* and can be represented in the SKOOL language (http://agrum.lip6.fr).

Throughout this paper, we will use an analogy with oriented-object programming in order to ease the presentation of our framework. It is organized as follows: after briefly introducing the classical PRM framework, we define the notions of attribute typing and type inheritance. Then we extend the notion of class inheritance with interfaces, to conclude

Pp. 273–281 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.



(a) A Bayesian network. The gray areas do not belong to the BN specification

(b) Two connected classes $\mathcal{E}$ and $\mathcal{F}$.

(c) The system declaration and the instance diagram corresponding to the BN of figure 1(a).

Figure 1: Representation of a BN as a PRM: analysis of the BN (a) reveals the use of two recurrent patterns, which are confined in two classes (b). Hence, a system equivalent to the BN may be built (c).

our contribution with the mechanisms for attribute and reference overloading. Finally we describe how PRMs with strong object-orientedness can be described using parfactors.

1 Description of PRMs

Fig. 1(a) shows a Bayesian Network (BN) encoding relations between two different kinds of patterns (variables X_i, Y_i on one hand and U_j, V_j, W_j on the other hand). We assume that the conditional probability tables (CPT) associated with variables with the same capital names are identical. When using PRMs, the main idea is to abstract each pattern as a generic entity, called a *class*, which encapsulates all the relations between the variables of the pattern. So, in Fig.1(b), $\mathcal{E}$ encapsulates precisely variables X_i and Y_i as well as their probabilistic relations (arc (X_i, Y_i)) and conditional probability distributions. The pattern of variables U_j, V_j, W_j cannot be directly encapsulated in a class since the CPTs of variables U_j are conditional to some variables Y_k (e.g., the CPT of U_3 is $P(U_3|Y_2)$ according to Fig.1(a)). Hence classes must have a mechanism allowing to refer to variables outside the class. In PRMs, this mechanism is called a *reference slot*. Basically, the idea is to create some function ρ connecting two classes and allowing both classes to access the variables of the other class. Now, as shown in Fig.1(c), the original BN can be built up from the PRM: it is sufficient to create two instances, say e_1 and e_2, of class $\mathcal{E}$ as well as three instances f_1, f_2, f_3 of $\mathcal{F}$ and connect them using one edge per reference slot. Note that there is no limit to the number of times an instance can be referenced (see e_1 in Fig.1(c)).

1.1 PRM-related definitions

In this section, we present the minimal set of definitions needed for the rest of the paper. The reader may refer to (Pfeffer, 2000) and (Getoor et al., 2007) for a more detailed presentation.

Definition 1 (Class)**.** A *class* $\mathcal{C}$ is defined by a Directed Acyclic Graph (DAG) over a set of attributes, i.e. random variables, $\mathbf{A}(\mathcal{C})$, a set of references (slots) $\mathbf{R}(\mathcal{C})$, and a probability distribution over $\mathbf{A}(\mathcal{C})$. To refer to a given random variable X (resp. reference ρ) of class $\mathcal{C}$, we use the standard Object Oriented notation $\mathcal{C}.X$ (resp. $\mathcal{C}.\rho$).

Definition 2 (Instance)**.** An *instance* c is the use (the instantiation) of a given class $\mathcal{C}$ in a BN. There are usually numerous instances of a given class $\mathcal{C}$ in a BN. Notation $c.X$ (resp. $c.\rho$) refers to the instantiation of $\mathcal{C}.X \in \mathbf{A}(\mathcal{C})$ (resp. $\mathcal{C}.\rho \in \mathbf{R}(\mathcal{C})$) in c. By abuse of notation, we denote the sets of such instantiations as $\mathbf{A}(c)$ and $\mathbf{R}(c)$ respectively.

Fig. 1(b) shows two classes, $\mathcal{E}$ and $\mathcal{F}$, with attributes $\mathcal{A}(\mathcal{E}) = \{X,\ Y\}$ and $\mathcal{A}(\mathcal{F}) = \{U,\ V,\ W\}$. There is also one reference in class $\mathcal{F}$ denoted by ρ which is used to define the dependencies between $\mathcal{E}.Y$ and $\mathcal{F}.U$. Such dependency is defined using a path, called a reference chain, from one attribute to another. In Fig. 1(b), the path representing the dependency between $\mathcal{E}.Y$ and $\mathcal{F}.U$ is $\mathcal{F}.\rho.Y$. More

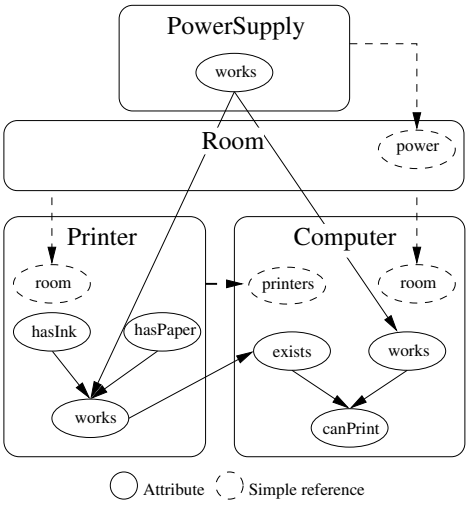


Figure 2: The printer example.

simply, $\rho.Y$ is said to be a parent of U in class $\mathcal{F}$.

Definition 3 (System, grounded net)**.** A system $\mathcal{S}$ is the representation of a BN as a set of class instances in which each reference has been linked to another instance. Conversely, the grounded network of a system $\mathcal{S}$ is the BN represented by $\mathcal{S}$.

As a consequence, in a system, each random variable $c.X$ is a copy of a random variable $\mathcal{C}.X \in \mathbf{A}(\mathcal{C})$ and is assigned a copy of the CPT assigned to $\mathcal{C}.X$ in $\mathcal{C}$. The difference between a system and a grounded net is that all structural information (classes, instances, references, ...) are lost when reasoning with a grounded net. Finally, PRMs are considered as an object-oriented formalism due to the encapsulation of attributes inside their classes. This feature is inherited from Object-Oriented Bayes Nets. Exploiting this encapsulation is the core of structured inference (Pfeffer, 2000).

1.2 Real Object-Oriented PRMs

The following discussion provides insight about how PRMs lack fundamental concepts of the object-oriented paradigm and how such concepts can greatly improve the representative power of PRMs. We will illustrate our point with a simple example of a printer breakdown diagnosis illustrated in Fig. 2. We consider a network with a power supply, black & white printers, color printers and computers. Printer's types, brands and ages vary from one printer to another. Printers and computers are placed in rooms and each computer is connected to every printer in the same room. All printers and computers are connected to the same power supply. Our main objective is to answer the following query: using a given computer, can I print? We can also think of other queries, not asked by a user but rather by an intervening technician: is a paper jam responsible for the printer's breakdown? Is the magenta cartridge of a color printer empty? Etc.

From the computer point of view, our system needs to take into account: (i) the fact that a printer prints in color is irrelevant for black & white printings; (ii) breakdowns can have different causes, but we only need to know whether printing is possible or not. From the technician perspective, we shall consider that: (i) different printers have different types of breakdowns, which can sometimes be partial, e.g. a color printer with no more cyan ink can still print in black & white; (ii) different types or brands imply different probabilities of breakdowns; (iii) the printer's features shall be taken into account since specific queries can be asked for each printer, e.g., can I print in color? Is the A3-tray empty? Etc.

These points of views force our system to be both generic (the computer's perspective) and specific (the technician's perspective). This is precisely why a strong object-oriented framework is needed. Let us do an analogy with computer programming. A class defines general concepts common to a family of objects. It is possible to define new concepts using inheritance: if class B inherits from class A, it inherits A's properties but can also specialize the concepts represented by A. Either by overloading A's attributes and methods (behavior specialization) or by adding new attributes and methods (functionality specialization). The next section proposes an extension of PRMs which will serve as a basis to strengthen class inheritance and we will show that this can be done with small and intuitive changes.

2 Attribute typing and type inheritance

Attribute typing arises naturally when using PRMs as a modeling framework: similarly to classes that represent repeated patterns in a system, an attribute type describes a family of random variables sharing the same domain. For instance, types *Boolean* and *state* would be the types of all the random variables

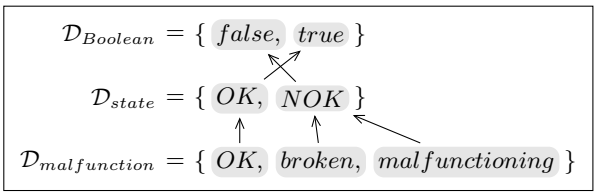


Figure 3: An illustration of type inheritance with attribute types *Boolean*, *state* and *malfunction*.

with domains $\{false, true\}$ and $\{OK, NOK\}$, respectively.

Definition 4 (Attribute typing)**.** An attribute type τ describes a family of distinct discrete random variables sharing the same domain $\mathcal{D}_\tau = \{l_1, \ldots, l_n\}$, where n is the domain size of τ.

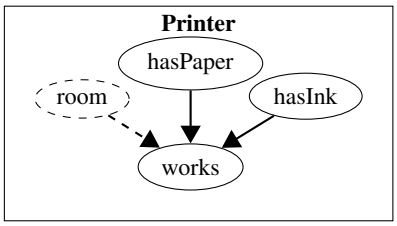
Types such as *Boolean* and *state* are frequently encountered when dealing with experts. For instance, they can be used to describe the states of equipments subject to breakdowns. In this case, type *state* enables a finer description of the possible failures than just the OK/NOK state. This can prove critical for some industrial applications: consider an air conditioner in a computer server room working improperly; then, assigning it state *malfunction* may help diagnose the servers malfunctions. Type *state* can be viewed as a *specialization* of type *Boolean*. Specializing general concepts into more specific ones is the goal of *inheritance*. Type inheritance is the process of decomposing labels into a partition of more specific and precise descriptions of a domain. To properly define this concept, we will need an additional notion, that of Domain Generalization Function (DGF):

Definition 5 (Domain Generalization Function)**.** A Domain Generalization Function (DGF) is a surjective function $\Phi : \mathcal{D}_\tau \rightarrow \mathcal{D}_\lambda$ where τ and λ are two distinct attribute types.

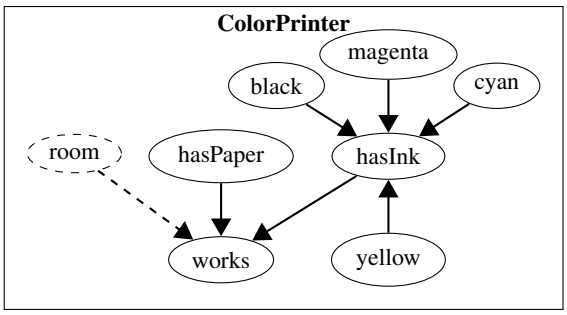
Obviously, given two distinct attribute types τ and λ, there exists a DGF $\Phi : \mathcal{D}_\tau \rightarrow \mathcal{D}_\lambda$ if and only if $|\mathcal{D}_\tau| \geq |\mathcal{D}_\lambda|$. DGFs will be used to define type inheritance in PRMs:

Definition 6 (Type inheritance)**.** An attribute type τ inherits from another attribute type λ if it is defined using a DGF $\Phi : \mathcal{D}_\tau \rightarrow \mathcal{D}_\lambda$.

Fig.3 illustrates type inheritance: in this figure, arcs represent the specialization of concepts. For



(a) Dependencies of the *Printer* class.



(b) Dependencies of the *ColorPrinter* class, which is a subclass of *Printer*.

Figure 4: Example of class inheritance. Dashed arcs represent dependencies with attributes in another class.

instance, attribute type *malfunction* has two labels, *broken* and *malfunction*, which are specializations of label *NOK* of attribute type *state*. As is, attribute inheritance is only a semantic relation: *state*'s label *OK* is a sort of *true*, *broken* is a sort of *false*, etc. We will show how to exploit such concepts probabilistically in the following sections.

3 Classes and interfaces

As in oriented-object programing, class inheritance in PRMs starts by a copy of the super class into its subclass. This implies that all attributes, references, dependencies, i.e. arcs, and CPTs are copied into the subclass. However, the content of the super class is only a basis for the subclass, as new attributes, references and dependencies can be added to the inherited structure. The first definitions of class inheritance for probabilistic models can be found in (Koller and Pfeffer, 1997) and (Bangsø and Wuillemin, 2000b). Note that these definitions differ greatly. In this paper, we propose some extensions of the work by Bangsø and Wuillemin.

3.1 Class inheritance

Fig. 4 illustrates class inheritance on the printer example of Fig. 2. Here, we introduced a new class, namely *ColorPrinter*, which is a subclass of *Printer*. Fig. 4(b) is a representation of the *ColorPrinter* class dependencies. This example suggests several remarks: (i) all the attributes and references belonging to class *Printer* also belong to *ColorPrinter*; (ii) new attributes have been added; (iii) attribute *ColorPrinter.hasInk* has additional parents (and thus a new CPT).

The first remark is similar with oriented-object programming languages: a subclass inherits the attributes and references of its super class. This implies that when an element is not overloaded, it is not necessary to redeclare it. The second remark is the functionality specialization of class inheritance: by adding new attributes, a subclass becomes more specific and offers new possibilities for entering evidence and submitting queries. In Fig. 4(b), attributes *black*, *magenta*, *cyan* and *yellow* represent the different kinds of inks used in a color printer, a feature that is not necessarily present in all printers. The third and fourth remarks are examples of attribute overloading, which consist of: (i) enabling changes in the values of the attribute's CPTs; (ii) adding or removing parents; (iii) overloading an attribute's type (this point is explained below).

3.2 Interface implementation

In modern programming languages, interfaces are used to handle multiple inheritance and to manipulate objects at a high abstract level. They define a set of methods which are guaranteed to exist in any class implementing them. Note that interfaces do not provide the bodies (the execution codes) of these methods but only their signatures. An interface in a PRM follows the same principle: it is a set of attributes and references; it defines neither probabilistic dependencies nor CPTs. As in programming languages, a PRM interface cannot be instantiated.

A PRM interface can be used to define dependencies between classes using abstraction: given two classes X and Y, if Y has an attribute depending on an attribute of X, then the only information needed is the type of X's attribute. As a consequence, the minimal set of information required to define probabilistic dependencies is composed of references and attribute's types.

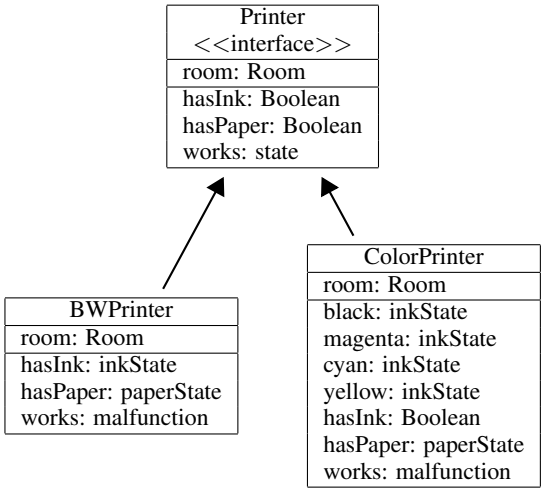


Figure 5: Two implementations of an interface.

Fig. 5 shows an example of an interface implementation, where the two classes *BWPrinter* and *ColorPrinter* implement interface *Printer* (which is no longer a class for this example). The *Printer* interface defines the minimal set of attributes and references any printer must declare: a reference to its room, whether it has ink, paper and whether it is working.

Fig. 5 is an alternative representation of classes using a UML syntax. Such syntax is necessary to point out attribute's and reference's types. It is more concise than the traditional representation of PRMs, i.e., the class dependency graph, see (Getoor et al., 2007). As already said, when creating a class, there is no need to know the dependency structure of the other classes to which it is related: only attribute and reference types are necessary for this task.

3.3 Multiple inheritance

Multiple inheritance is one of the major issues when defining an object-oriented formalism. The problem arises when diamond-shaped inheritance appears, as illustrated in Fig. 6. An ambiguity results from how the properties of class A are inherited by class D since two distinct paths exist from A to D (through B or C). Furthermore, if some properties of A are overloaded in B and C, which one should be inherited by D? Such issue can be dealt by using inter-

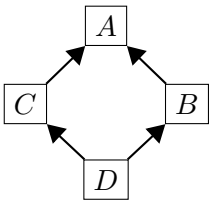


Figure 6: A diamond-shaped inheritance graph.

<state>works	*OK*	*Broken*	*malfunction*
OK	1	0	0
NOK	0	1	1

Table 1: The CPT of *BWPrinter.works* cast descendant which is of type *state*.

faces: since an interface only declares the existence of properties, each class implementing a given interface must declare and define itself those properties. The major drawback of this approach is that there is no reuse of properties definitions, i.e. there is code duplication. Another solution consists of explicitly declaring from which superclass a given property is inherited. But this proves to be cumbersome and bug prone. For this reason, we chose to use the interface-based solution. In addition, the notion of an interface is well suited for the PRM framework. Note that a class can implement as many interfaces as it designer wants to.

4 Attribute and reference overloading

In object-oriented programming languages, overloading is used to modify inherited properties. This is exactly what PRM attribute overloading and reference overloading do. In section 3.1, we showed how attribute overloading could be performed using inheritance. Now, by adding attribute typing, it becomes possible to also overload attribute's types.

4.1 Type overloading

People familiar with PRMs will remark that, in Fig. 5, if a class has a dependency over *Printer.works* it expects an attribute of type *state* and defines its conditional probability tables accordingly. However, connecting such a class to an instance of *BWPrinter* results in an incoherent probability distribution since the attribute referenced is of type *malfunction*. To fix this kind of problem we need the concept of *cast descendants*.

Cast Descendants are automatically generated attributes which are used to cast beliefs of an attribute into one of its super type. By exploiting Domain Generalization Functions (DGFs), it is possible to use deterministic tables to obtain beliefs with the correct domain size. Tab. 1 shows the conditional probability table of *BWPrinter.works* cast descendant, which casts type *malfunction* into type *state*. Algorithm 1 illustrates more formally how cast descendants are generated. The *goal* variable is the overloaded type and *a* the attribute whose type is a subtype of *goal*. The algorithm simply adds children to *a* until the goal type is reached. Procedure *generateCastCPT()* uses DGFs to generate deterministic tables as shown in Table 1.

```
Data: Type goal, Attribute a
Type t = a.type;
Attribute current = a;
while t ≠ goalType do
    Attribute child = new Attribute();
    child.type = t.super;
    child.cpt = generateCastCPT(t.super, t);
    current.addChild(child);
    current = child;
    t = t.super;
end
```

Algorithm 1: Cast descendant generation.

4.2 Reference overloading and instantiation

As seen previously, it is possible to define references in classes as well as in interfaces. We have shown how interfaces can be used to define probabilistic dependencies and since we introduced class inheritance and interface implementation, we can obviously use reference overloading. Given two classes X and Y, if Y is a subclass of X and if there exists a reference ρ in X referencing a class Z (or an interface I), then Y can overload ρ with a reference referencing any subclass of Z (or referencing any implementation of I).

Instantiating a reference amounts to linking it to an instance of the correct class in a given system. Given an instance x of class X and a reference $x.\rho$ referencing a class Z (or an interface I) $x.\rho$ can be instantiated in any instance of a subclass of Z (or any instance of a class implementing I). Class inheritance, interface implementation and cast descendants guarantee the existence of attributes de-

fined in a class (or interface) in any of its subclass (or implementation), which is sufficient to ensure a coherent probabilistic distribution.

5 Parfactor representation of PRMs

A large part of the statistical relational learning community has chosen first-order probabilistic models as their main framework. Actually, the only exact inference algorithm for first-order probabilistic models is lifted probabilistic inference (de Salvo Braz et al., 2005) and (Brian et al., 2008). Parfactors are the common formalization used in these approaches. It is important to note that, like most first-order probabilistic models, parfactors are more generic than PRMs: they can be used to represent complex systems impossible to represent using PRMs. However, they are less suited for modeling large-scale systems. Hence it is useful to be able to express PRMs in such a formalism. We will give a short definition of parfactors, or parametric factors, as they are given in (Poole, 2003).

Definition 7 (Parfactor)**.** A parfactor is a triple $\langle C, V, t\rangle$ where C is a set of constraints on parameters, V is a set of parametrized random variables and t is a table representing a factor from random variables of V to $\Re^+$.

Algorithm 2 details formally how an attribute can be converted into a set of parfactors. We will detail this algorithm using attribute *ColorPrinter.works* of Fig. 4(b). Since we represent PRMs, a parfactor's table will always be the conditional probability table of an attribute. Classes and instances are represented as parameters of parametric random variables, which are the equivalent of attributes in the PRM formalism. To ensure the exact representation of the structure encoded by the classes of a PRM, it is necessary to use two different types of constraints.

```
Input: Class c, Attribute attr
Output: Parfactor fctr
Parfactor fctr;
Add a isA() constraint over c's type;
Add a parametrized variable named by attr and prefixed
by attr's type;
foreach parent prt of attr do
    if prt not in A(c) then
        Add a isA() constraint over prt's class type;
        foreach reference ρ in the slot chain from attr to
        prt do
            Add a relational constraints in fctr
            matching ρ;
            Add a isA() constraint over ρ range type;
        end
    end
    Add a parametrized variable named by prt and
    prefixed by prt's type;
end
Copy in fctr's table attr's CPT;
return fctr
```

Algorithm 2: Parfactor generation of an attribute.

To represent classes, class inheritance and interface implementation we will use *isA()*-like constraints (e.g. *isAPrinter(X)*). Relations can be expressed as binary constraints in which each parameter has a *isA()* constraint (e.g. $room(X, R) \wedge isAPrinter(X) \wedge isARoom(R)$). The *ColorPrinter.works* attribute in Fig. 4 can be represented by the following parfactor:

$$\langle \{isAColorPrinter(X) \wedge isARoom(Y) \wedge isAPowerSupply(Z) \wedge room(X,Y) \wedge power(Y,Z)\}, \{malfunction\_works(X),\ paperType\_hasPaper(X),\ Boolean\_hasInk(X),\ state\_works(Z)\},\ t\ \rangle$$

The first part of this parfactor is composed of type constraints (*isAColorPrinter()*, *isARoom()* and *isAPowerSupply()*) and relational constraints (*room()* and *power()*). The second part contains the dependencies of the parfactor, which are the parametrized random variables *malfunction_works(X)*, *paperType_hasPaper(X)*, *Boolean_hasInk(X)* and *state_works(Z)*. The Cartesian product of their values is mapped to the values in *t*, which represent the CPT of *ColorPrinter.works*.

The *isA()* constraints encode the inheritance scheme of a PRM if, for each instance i of a system, a grounded variable is declared for each type of i, i.e. for all of its super classes and implemented interfaces. For example, an instance *coloria* of the *ColorPrinter* class will be represented with the following grounded variables: *isAColorPrinter(coloria)* and *isAPrinter(coloria)*.

Finally, cast descendants can be represented by including types names in the parametric variables declarations. For example *malfunction_works(X)* stands for the attribute *ColorPrinter.works* of type *malfunction*. Then, by generating parfactors for each cast descendant, the constraints names will ensure the correct structure. For example, the cast descendant *ColorPrinter.works* will be declared as:

$\langle \{isAColorPrinter(X)\}$
$\{state\_works(X),\ malfunction\_works(X)\},\ t \rangle$

At first sight, such a representation seems cumbersome but it illustrates the expressive power of parfactors and of first-order probabilistic models. First-order logic can be used to express very complex relations: only two types of constraints are necessary to represent all the notions presented in this paper. However such expressive power has a major flaw as semantics and relations are hidden in the mass of constraints declarations. When dealing with large-scale systems, creating and maintaining such knowledge base can be extremely difficult. PRMs with the strengthened object-oriented aspect we proposed here are a proposition to manage such knowledge with a formalism less expressive but much more scalable.

6 Conclusion

We proposed a strong object-oriented representation of PRMs by introducing interfaces, attribute typing, type inheritance, attribute and reference overloading. Such notions strengthen the expressive power of PRMs when dealing with structured and known systems. In addition, we have shown how PRMs with these features can easily be represented as parfactors, closing a gap between PRMs and more recent first-order probabilistic models. Strengthening the object-oriented features of PRMs enables a better representation of complex systems as well as the creation of new models in fields such as troubleshooting, reliability and risk management, where such models were often difficult to represent until now. Parfactors are used in the state-of-the-art lifted probabilistic inferences. Enabling the expression of PRM models into such formalisms will help comparing different first-order probabilistic implementations. However there is still room for improvements, especially for the graphical representation of PRMs and the implementation of user-friendly tools for model design and maintenance. Finally, the perspective of exploiting hierarchical knowledge can lead to new inference algorithms in PRMs.

Acknowledgments: this work has been supported by the DGA and has benefited comments, suggestions and ideas from the SKOOB consortium members (http://skoob.lip6.fr).

References

O. Bangsø and P.-H. Wuillemin. 2000a. Top-down construction and repetitive structures representation in Bayesian networks. In *Proc. of FLAIRS 2000*, pages 282–286.

Olav Bangsø and Pierre-Henri Wuillemin. 2000b. Object Oriented Bayesian Networks: A framework for topdown specification of large Bayesian networks and repetitive structures. Technical report, Department of Computer Science, Aalborg University., Aalborg, Denmark.

Milch Brian, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. 2008. Lifted probabilistic inference with counting formulas. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1062–1068.

R. de Salvo Braz, E. Amir, and D Roth. 2005. Lifted first- order probabilistic inference. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1319–1325.

L. Getoor and B. Taskar. 2007. *Introduction to Statistical Relational Learning*. The MIT Press.

Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. 2007. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *An Introduction to Statistical Relational Learning*. MIT Press.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models*. The MIT Press.

D. Koller and A. Pfeffer. 1997. Object-oriented Bayesian networks. In *Proceedings of the 13th Annual Conference on Uncertainty in AI*, pages 302–313.

K.B. Laskey. 2008. MEBN: A language for first-order Bayesian knowledge bases. *Artificial Intelligence*, 172:140–178.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman.

A.J. Pfeffer. 2000. *Probabilistic Reasoning for Complex Systems*. Ph.D. thesis, Stanford University.

David Poole. 2003. First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 985–991.

Xiang Yang. 2002. *Probabilistic Reasoning in Multi-Agent Systems: A Graphical Models Approach*. Cambridge University Press.

Acquisition and Computation Issues with NIN-AND Tree Models

Yang Xiang
University of Guelph, Canada

Abstract

Most techniques to improve efficiency of conditional probability table (CPT) acquisition for Bayesian network (BN) can only represent reinforcing causal interactions. The non-impeding noisy-AND (NIN-AND) tree is the first causal model that explicitly expresses reinforcement, undermining, and their mixture, while its acquisition is of linear complexity. We address three issues on acquisition and computation with these models. In particular, we propose methods to improve computation of conditional probability from a model, to improve the efficiency of CPT computation from these models, and to address NIN-AND tree acquisition by elicitation of pairwise causal interactions.

1 Introduction

To acquire a BN, for each non-root node, a CPT needs to be specified. When the BN is constructed along the causal direction, a CPT is the distribution of an effect conditioned on its n causes. In general, the complexity of CPT assessment is exponential on n. A number of techniques have been proposed to make the assessment more efficient. Noisy-OR (Pearl, 1988) is the most well known that reduces this complexity to linear. A number of extensions have also been proposed such as (Heckerman and Breese, 1996; Galan and Diez, 2000; Lemmer and Gossink, 2004). However, noisy-OR, noisy-AND (Galan and Diez, 2000), as well as related techniques, can only represent causal interactions that are reinforcing (Xiang and Jia, 2007).

The NIN-AND tree (Xiang and Jia, 2007) is a recently proposed technique for efficiently modeling and acquiring CPT.[1] It inherits the reinforcing behavior of noisy-OR (Pearl, 1988). It inherits some features of noisy-AND (Galan and Diez, 2000) while moves away from its impeding behavior (see (Xiang and Jia, 2007)) to allow modeling of undermining. It also inherits the flexibility of recursive noisy-OR (Lemmer and Gossink, 2004) to allow probabilities of multi-causal events as input. As the result, NIN-AND tree provides the first causal model that explicitly expresses reinforcing and undermining causal interactions, as well as their mixture. Its acquisition involves elicitation of probability parameters whose number is linear on n, and a tree topology whose size is linear on n.

This paper addresses three technical issues regarding acquisition of NIN-AND tree models and computations performed on them. First, given an NIN-AND tree model for an effect and some of its present causes, the probability of the effect can be computed, conditioned on that these causes are present and the remaining causes are absent . We propose a new algorithm to compute this probability from an NIN-AND tree model, that improves upon the alternative in (Xiang and Jia, 2007). The new algorithm takes advantage of minimal NIN-AND tree models and enables more efficient CPT computation.

Second, according to a method in (Xiang and Jia, 2007), the exponential number of probability parameters in a CPT can each be computed from a suitable NIN-AND tree model. Although all such models can be obtained by modifying a common base model, an exponential number of alternative NIN-AND tree models must be created in the process. We propose a new method that can directly compute all probability parameters of a CPT from the base model, sav-

[1] Being unaware of this work and its precursor (Xiang and Jia, 2006), (Maaskant and Druzdzel, 2008) independently presented special cases of NIN-AND tree models.

Pp. 281–289 in *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010)*, edited by Petri Myllymäki, Teemu Roos and Tommi Jaakkola. HIIT Publications 2010–2.

ing the computation of creating the exponential number of NIN-AND trees.

Third, acquisition of the tree topology is a critical step in specifying an NIN-AND tree model. Three methods have been proposed: direct specification (Xiang and Jia, 2007), two-step menu selection (Xiang et al., 2009b), and identification by pairwise causal interaction (Xiang et al., 2009a). An expert-specified pairwise causal interaction function may have no corresponding NIN-AND tree model. We propose a technique to address this issue.

In this paper, we focus on binary effect and causes. For generalization of NIN-AND tree models to multi-valued effect and cause variables, see (Xiang, 2010).

The remainder of the paper is organized as follows: The background on NIN-AND tree models is introduced in Section 2. Computation of conditional probability from an NIN-AND tree model is addressed in Section 3. The method for computing CPT without exponential generation of NIN-AND trees is presented in Section 4. How to facilitate expert in NIN-AND tree acquisition by pairwise causal interaction is described in Section 5.

2 Background

This section is mostly based on (Xiang and Jia, 2007). An *uncertain cause* is a cause that can produce an effect but does not always do so. Denote a binary effect variable by e and a set of binary cause variables of e by $X = \{c_1, ..., c_n\}$. Denote $e = true$ by e^+ and $e = false$ by e^-. Similarly, for each cause c_i, denote $c_i = true$ by c_i^+ and $c_i = false$ by c_i^-. Denote the set of *all causes* (including a leaky cause) of e by C.

A *singular causal event* refers to an event that a cause c_i caused its effect e to occur successfully when all other causes of e are absent. Denote this causal event by $e^+ \leftarrow c_i^+$ and its probability by $P(e^+ \leftarrow c_i^+)$. The singular causal failure event, where e is false when c_i is true and all other causes of e are false, is denoted by $e^+ \not\leftarrow c_i^+$. Denote the multi-causal event that a set $X = \{c_1, ..., c_n\}$ $(n > 1)$ of causes caused e by $e^+ \leftarrow c_1^+, ..., c_n^+$ or $e^+ \leftarrow \underline{x}^+$.

Causes reinforce each other if collectively they are at least as effective in causing the effect as some acting by themselves. If collectively they are less effective, then they undermine each other. For $C = \{c_1, c_2\}$, if c_1 and c_2 undermine each other, all the following hold:

$$P(e^+|c_1^-, c_2^-) = 0, P(e^+|c_1^+, c_2^-) > 0, P(e^+|c_1^-, c_2^+) > 0,$$
$$P(e^+|c_1^+, c_2^+) < min(P(e^+|c_1^+, c_2^-), P(e^+|c_1^-, c_2^+)).$$

Reinforcement and undermining occur between individual as well as sets of variables. Variables within each set can be reinforcing, while the sets can undermine each other. Hence, W_i in Def. 1 is not necessarily a singleton.

Definition 1. *Let $R = \{W_1, W_2, ...\}$ be a partition of a set X of causes, $R' \subset R$ be any proper subset of R, and $Y = \cup_{W_i \in R'} W_i$. Sets of causes in R reinforce each other, iff* $\forall R'\ P(e^+ \leftarrow \underline{y}^+) \leq P(e^+ \leftarrow \underline{x}^+)$. *Sets of causes in R undermine each other, iff* $\forall R'\ P(e^+ \leftarrow \underline{y}^+) > P(e^+ \leftarrow \underline{x}^+)$.

Disjoint sets of causes $W_1, ..., W_m$ satisfy *failure conjunction* iff $e^+ \not\leftarrow \underline{w}_1^+, ..., \underline{w}_m^+ = \wedge_{i=1}^m (e^+ \not\leftarrow \underline{w}_i^+)$. That is, collective failure is attributed to individual failures. They also satisfy *failure independence* iff $P(\wedge_{i=1}^m (e^+ \not\leftarrow \underline{w}_i^+)) = \prod_{i=1}^m P(e^+ \not\leftarrow \underline{w}_i^+)$. Disjoint sets of causes $W_1, ..., W_m$ satisfy *success conjunction* iff $e^+ \leftarrow \underline{w}_1^+, ..., \underline{w}_m^+ = \wedge_{i=1}^m (e^+ \leftarrow \underline{w}_i^+)$. That is, collective success requires individual effectiveness. They also satisfy *success independence* iff $P(\wedge_{i=1}^m (e^+ \leftarrow \underline{w}_i^+)) = \prod_{i=1}^m P(e^+ \leftarrow \underline{w}_i^+)$.

Causes are reinforcing whenever they satisfy failure conjunction and independence, and they are undermining whenever they satisfy success conjunction and independence. Hence, undermining can be modeled by a direct NIN-AND gate (Fig. 1, left), and reinforcement by a dual NIN-AND gate (right). Complex mixture of reinforcement and undermining can be modeled by an NIN-AND tree defined below.

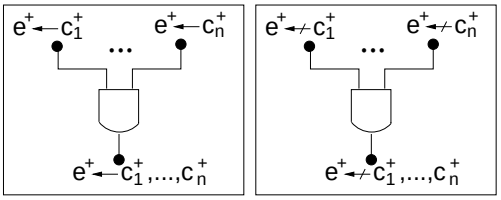


Figure 1: Direct (left) and dual (right) NIN-AND gates

Definition 2. *An* `NIN-AND tree` T *is a directed tree for effect* e *and a set* $X = \{c_1, ..., c_n\}$ *of occurring causes.*

1. *There are two types of nodes,* `event` *nodes (a black oval of in-degree* ≤ 1 *and out-degree* ≤ 1*) and* `gate` *nodes (a NIN-AND gate) of in-degree* ≥ 2 *and out-degree* 1.

2. *There are two types of links, each connecting an event and a gate along input-to-output direction of gates,* `forward` *links (a line) and* `negation` *links (with a white oval at gate end).*

3. *Each terminal node is an event labelled by a causal event* $e^+ \leftarrow \underline{y}^+$ *or* $e^+ \not\leftarrow \underline{y}^+$*. There is a single* `leaf` *(no child) with* $\underline{y}^+ = \underline{x}^+$*, connecting to the* `leaf gate`*. For each* `root` *(no parent; indexed by i),* $\underline{y}_i^+ \subset \underline{x}^+$*,* $\underline{y}_j^+ \cap \underline{y}_k^+ = \emptyset$ *for* $j \neq k$*, and* $\bigcup_i \underline{y}_i^+ = \underline{x}^+$*.*

4. *For inputs to a direct gate* g*, each is either connected by a forward link to a node labelled* $e^+ \leftarrow \underline{y}^+$*, or by a negation link to a node labelled* $e^+ \not\leftarrow \underline{y}^+$*. Gate* g *outputs by a forward link to a node labelled* $e^+ \leftarrow \cup_i \underline{y}_i^+$*.*

5. *For inputs to a dual gate* g*, each is either connected by a forward link to a node labelled* $e^+ \not\leftarrow \underline{y}^+$*, or by a negation link to a node labelled* $e^+ \leftarrow \underline{y}^+$*. Gate* g *outputs by a forward link to a node labelled* $e^+ \not\leftarrow \cup_i \underline{y}_i^+$*.*

Fig. 2 shows an NIN-AND tree, where $X = \{c_1, ..., c_5\}$. The leaf gate g_1 is dual, and so is g_3. The remaining gates are direct. Causes c_2 and c_5 are undermining each other, but they reinforce c_3 (and vice versa). Collectively, the three undermines c_4. The four of them reinforces c_1.

A root node may be labelled by a singular or multi-causal event. In this paper, we assume that it is singular. When we refer to a node in T by v, we overload the symbol v to refer also to the causal event that labels the node.

Definition 3. *An NIN-AND tree model* M *is a quadruple* (X, e, T, B)*.* X *is a set of uncertain causes of effect* e*.* T *is an NIN-AND tree for* e *and* X*.* B *is a set of parameters, one for each root in* T *and being a potential in* $(0, 1]$ *over the corresponding causal event.*

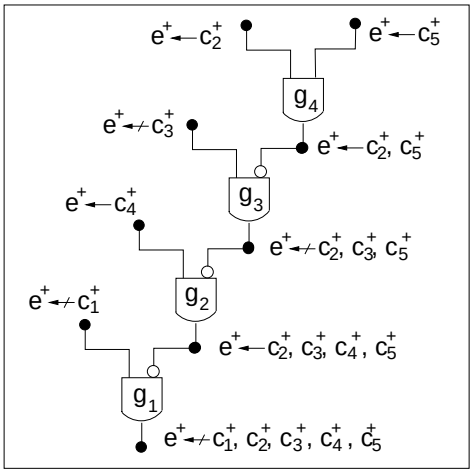


Figure 2: An NIN-AND tree

For a root v in T, when parameter $B(v) < 1$, it represents probability $P(v)$. When $B(v) = 1$, it plays a special role to be described below. T is a *full* NIN-AND tree if $X = C$, otherwise T is *partial*. Model M is defined to be full or partial similarly.

Model $M = (X, e, T, B)$ can be obtained by eliciting tree T plus $|X|$ singular causal probabilities. Probability $P(e^+|\underline{x}^+, \underline{y}^-)$, where $C = X \cup Y$ and $X \cap Y = \emptyset$, can then be derived as $P(e^+ \leftarrow \underline{x}^+)$.

Definition 4. *An NIN-AND tree* T *is* `minimal` *if, whenever a gate* g *feeds directly into another gate* g'*, the types (direct or dual) of* g *and* g' *differ.*

The NIN-AND tree in Fig. 2 is minimal. Model $M = (X, e, T, B)$ is minimal if T is minimal.

3 Probability of Leaf Causal Event

Algorithm 1 below computes probability of the leaf event of a minimal NIN-AND tree model M. All nodes mentioned are event nodes. We refer to the number of gate nodes from an event node v to the leaf as the *level* of v. If event node w feeds into a gate that connects to event node v, we simply refer to w as the parent of v.

Algorithm 1. *GetLeafEventProb(M)*
Input: A minimal NIN-AND tree model M*;*

1 *for each non-root node* v*, set* $B(v) = 1$*;*
2 $L =$ *max level of any root in* T*;*
3 *for* $lev = L - 1$ *to 0, do*
4 *for each non-root node* v *at level* lev*, do*
5 *for each parent* w *of* v*, do*
6 *if* w *is a root,* $B(v) = B(v) * B(w)$*;*
7 *else if* $B(w) < 1$*,*
8 $B(v) = B(v) * (1 - B(w))$*;*
9 *return* $B(x)$ *where* x *is the leaf;*

Consider model $M = (X, e, T, B)$, where T is shown in Fig. 2 and B is defined as follows:

$$P(e^+ \not\leftarrow c_1^+) = 0.2, \quad P(e^+ \leftarrow c_2^+) = 0.85,$$
$$P(e^+ \not\leftarrow c_3^+) = 0.3, \quad P(e^+ \leftarrow c_4^+) = 0.65,$$
$$P(e^+ \leftarrow c_5^+) = 0.75$$

GetLeafEventProb(M) returns $P(e^+ \not\leftarrow c_1^+, ..., c_5^+) = 0.0841375$, that is, $P(e^+ \leftarrow c_1^+, ..., c_5^+) = 0.9158625$.

The following proposition shows that if all parameters in M are root event probabilities, GetLeafEventProb(M) computes the leaf event probability exactly.

Proposition 1. *Let* M *be a minimal NIN-AND tree model with the leaf node* x *and with* $B(v) = P(v) < 1$ *for each root* v*. Then* $B(x)$ *returned from* $GetLeafEventProb(M)$ *is equal to* $P(x)$ *as determined by root event probabilities and causal interactions encoded in the tree topology of* M*.*

Proof: We prove by induction on the maximal level L of any root. If $L = 1$, all parents of the leaf x are roots. The loop beginning at line 3 iterates exactly once and so does the loop at line 4 (relative to x). Each parent w of leaf x has $B(w) = P(w)$ from specification of M. Hence, we have $B(x) = \prod_w P(w)$ from line 6. Since each $P(w) < 1$, we have $B(x) < 1$.

If the leaf gate g is direct, by Def. 2 (4), each w is a causal success and so is x. T implies that causes satisfy success conjunction and independence. Hence, $B(x)$ correctly models their undermining relation, and $B(x) = P(x)$. If g is dual, by Def. 2 (5), each w is a causal failure and so is x. T implies that causes satisfy failure conjunction and independence. Hence, $B(x)$ correctly models their reinforcing relation, and $B(x) = P(x)$.

Assuming that the proposition holds with $L \leq k$ for $k \geq 1$, consider the case $L = k + 1$. Let y be a node at level 1. If y is a root, we have $B(y) = P(y)$ from specification of M.

If y is a non-root, it defines a subtree T' of T with the leaf being y, and a corresponding submodel M'. The loop at line 3 iterates $k+1$ times during $GetLeafEventProb(M)$. The computation in the first k iterations that involves nodes in T' is exactly the same computation performed by $GetLeafEventProb(M')$. Since for M', $L' \leq k$, by inductive hypothesis, $B(y)$ returned by $GetLeafEventProb(M')$ is exactly $P(y)$. Hence, at the end of k'th iteration during $GetLeafEventProb(M)$, we have $B(y) = P(y) < 1$.

The above argument holds for each node y at level 1. At the $k+1$'th iteration (loop at line 3) of $GetLeafEventProb(M)$, node v is the leaf x. If y is a root, $B(y)$ affects $B(x)$ through line 6. This is correct since y and x are either both causal successes or both causal failures. If y is a non-root, since $B(y) < 1$, test in line 7 will succeed, and $B(y)$ affects $B(x)$ through line 8. This is correct since M is minimal. Either y is a causal failure and x is a causal success, or y is a causal success and x is a causal failure. Hence, $B(x)$ is computed by the correct product according to the type of leaf gate (and its implied causal interaction), and we have $B(x) = P(x)$. □

Computation of leaf event probability is an essential component of NIN-AND tree modeling. The algorithm above improves upon that in (Xiang and Jia, 2007). It takes advantage of a minimal model, while the latter does not. It is iterative while the latter is recursive. Hence, when the number of input events of a gate is upper bounded, its complexity is $O(n)$, while that of the latter is $O(n^2)$. This saving is amplified during CPT computation, as it needs to be preformed $O(2^n)$ times, as shown below.

4 CPT Computation without Model Regeneration

NIN-AND tree models can be used to acquire a CPT $P(e|C)$ efficiently. As proposed in (Xiang

and Jia, 2007), the model $M = (C, e, T, B)$ is elicited first, from which $P(e^+|\underline{c}^+)$ can be computed as $P(e^+ \leftarrow \underline{c}^+)$. For each $P(e^+|\underline{x}^+, \underline{y}^-)$, where $X \subset C$ and $Y = C \setminus X$, a model $M' = (X, e, T', B')$ is created by modifying T into T'. The modification involves removing roots corresponding to absent causes as well as downstream event and gate nodes, and arrives at a significantly different tree structure. $P(e^+|\underline{x}^+, \underline{y}^-)$ is then computed from M' as $P(e^+ \leftarrow \underline{x}^+)$. Hence, this method requires generation of $O(2^{|C|})$ NIN-AND tree models M'.

For instance, suppose $C = \{c_1, ..., c_5\}$ and T is as shown in Fig. 2. To compute $P(e^+|c_1^+, c_2^+, c_3^+, c_4^-, c_5^+) = P(e^+ \leftarrow c_1^+, c_2^+, c_3^+, c_5^+)$, T is modified by deleting event node $e^+ \leftarrow c_4^+$ and gate g_2 to create T' as shown in Fig. 3 (a). T' is not minimal. To use the equivalent minimal NIN-AND tree, the model shown in (b) needs to be created.

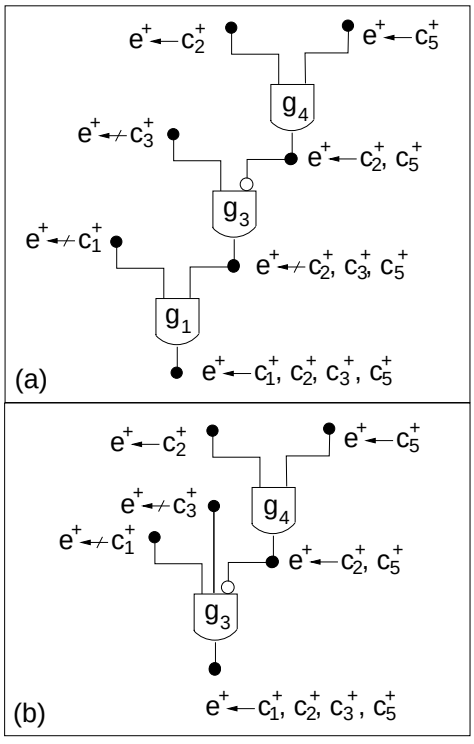


Figure 3: NIN-AND trees modified from Fig. 2 when c_4 is absent

We propose an alternative method below for CPT computation which does not require generation of an exponential number of models.

Algorithm 2. *GetCPTByNinAndTree(M)*
Input: A minimal, full NIN-AND tree model M;

```
1  M' = M;
2  P(e+|c+) = GetLeafEventProb(M');
3  P(e+|c-) = 0;
4  for each non-empty X ⊂ C, do
5     Y = C \ X;
6     for each root v in T, do
7        if causal event at v involves z ∈ X,
8           set B'(v) = B(v);
9        else set B'(v) = 1;
10    P(e+|x+, y-) = GetLeafEventProb(M');
11 return P(e+|C);
```

$GetCPTByNinAndTree(M)$ uses a duplicated model M' of M for computing all probabilities in $P(e^+|C)$. $P(e^+|\underline{c}^+)$, where all causes are present, is derived from unmodified M'. $P(e^+|\underline{c}^-)$, where all causes are absent, is trivially set, as C is the entire set of causes of e.

All other probabilities in $P(e^+|C)$ are in the form $P(e^+|\underline{x}^+, \underline{y}^-)$, and are computed by the loop started in line 4, one per iteration. In the iteration relative to a given $P(e^+|\underline{x}^+, \underline{y}^-)$, the potential for each root in M' is reset by the loop started in line 6. For each root whose causal event concerns the cause z, if $\underline{x}^+$ includes z, the potential associated with the root is set to the potential in the corresponding root in M. Otherwise, $\underline{y}^-$ must include z, and the potential of the root is set to 1. We consider the effect of this assignment below.

According to Algorithm 1, root potentials are not modified. From line 6 of Algorithm 1, if the potential of a root v is 1, it has no impact to the potential of its child node, and hence has no impact to the value of $P(e^+|\underline{x}^+, \underline{y}^-)$. When a cause z is included in $\underline{y}^-$, it is absent. Hence, this is exactly the effect expected.

Furthermore, consider the child node u of the root v. If all its parents of u are roots like v, $B(u)$ will remain at the value 1 due to line 1 of Algorithm 1. Hence, $B(u)$ will have no impact to the value of $P(e^+|\underline{x}^+, \underline{y}^-)$. This is exactly the effect expected when causes appearing upstream to u are all absent.

On the other hand, if any root parent w of

u involves a cause z that is included in $\underline{x}^+$, it must be the case $B(w) < 1$ (Algorithm 2, line 8). By line 6 of Algorithm 1, $B(w)$ affects $B(u)$ correctly as a factor.

By an inductive argument, any non-root w with ancestors that involve causes in $\underline{x}^+$, it must be the case $B(w) < 1$. By lines 7 and 8 of Algorithm 1, $B(w)$ will affect $B(u)$ correctly as a factor, where u is the child of w. This leads to the following theorem whose proof can be phrased based on the above analysis.

Theorem 1. *Let M be a minimal, full NIN-AND tree model with $B(v) = P(v) < 1$ for each root v. Then $P(e^+|C)$ returned by GetCPT-ByNinAndTree(M) is exact relative to root event probabilities and causal interactions encoded in the tree topology of M.*

Note that $GetCPTByNinAndTree(M)$ needs only modify the root potentials of M' for the computation of each $P(e^+|\underline{x}^+, \underline{y}^-)$. Hence, the computation to generate an exponential number of NIN-AND tree models is saved without affecting exactness.

5 Tree Structure Acquisition by Pairwise Causal Interaction

To acquire a minimal NIN-AND tree model $M = (C, e, T, B)$, the tree structure T must be obtained. It has been shown (Xiang et al., 2009a) that T defines a *pairwise causal interaction* function *pci* from pairs of distinct causes $\{c_i, c_j\} \subset C$, where $i \neq j$, to the set $\{rif, udm\}$, where rif stands for reinforcing and udm stands for undermining. Table 1 shows an example.

Table 1: The *pci* function defined by NIN-AND tree in Fig. 2

c_i	c_j	$pci(c_i, c_j)$	c_i	c_j	$pci(c_i, c_j)$
c_1	c_2	rif	c_2	c_4	udm
c_1	c_3	rif	c_2	c_5	udm
c_1	c_4	rif	c_3	c_4	udm
c_1	c_5	rif	c_3	c_5	rif
c_2	c_3	rif	c_4	c_5	udm

Furthermore, for $|C| \leq 10$, it has been computationally verified (Xiang et al., 2009a) that, the *pci* function of T is unique among *pci* functions of alternative minimal NIN-AND trees over C and e. Therefore, given the *pci* function of T, the tree structure can be identified uniquely. For instance, given Table 1, Fig. 2 is the unique full minimal NIN-AND tree. This fact suggests that, instead of eliciting T directly from expert, it can be obtained alternatively by eliciting the corresponding *pci* function. Denote $n = |C|$, the number of pairwise causal interactions to be elicited is $n(n-1)/2$. We investigate this approach below.

First of all, does every *pci* function correspond to a NIN-AND tree? Since given $n = |C|$, the number of pairwise causal interactions is $n(n-1)/2$, the number of alternative *pci* function for n causes is $2^{n(n-1)/2}$.

On the other hand, every NIN-AND tree defines the same *pci* function as its minimal NIN-AND tree. Hence, it is sufficient to count the number of minimal NIN-AND trees. Unfortunately, no closed formula is known for the number of minimal NIN-AND trees over n causes. Instead, the number can be obtained by enumeration of minimal NIN-AND trees given n (Xiang et al., 2009b; Xiang et al., 2009a).

Table 2 compares the number of alternative *pci* functions and that of minimal NIN-AND trees given n. The rows are indexed by n. The second column lists the number of *pci* functions. The third column lists the number of minimal NIN-AND trees. The last column is the ratio of the number in the second column over that in the third column.

Table 2: Comparison of the number of *pci* functions and that of minimal NIN-AND trees

n	No. pci func	No. trees	Ratio
3	8	8	1
4	64	52	1.2
5	1024	472	2.2
6	32768	5504	6
7	2097152	78416	26.7
8	268435456	1320064	203
9	68719476736	25637824	2680

We refer to a *pci* function that can be imple-

mented by an NIN-AND tree as *feasible*. Otherwise, it is *infeasible*. Hence, Table 2 shows that other than $n = 3$, there are more infeasible *pci* functions than feasible ones. It follows that in acquisition of an NIN-AND tree by elicitation of the corresponding *pci* function, an expert could specify an infeasible *pci* function. This could happen for at least two reasons. The expert could misspecify some pairwise interaction (human error), or the specified function is not expressible by an NIN-AND tree model.

From the human error perspective, having more infeasible *pci* functions is desirable, because a misspecified *pci* function can thus be detected rather than being regarded as feasible and causing an unintended NIN-AND tree to be returned unnoticed.

Furthermore, as n grows, the ratio between number of *pci* functions and number of minimal NIN-AND trees grows very rapidly. This is also desirable because, as n grows, a misspecified *pci* function is even more likely to be detected. This counter-balances the increased chance of human error due to the increased number of pairwise causal interactions to be elicited as n grows.

Next, we consider what an NIN-AND tree acquisition software should do when expert specifies an infeasible *pci* function. Rejecting the function is a simple measure. However, it would be more helpful to assist expert in correcting the mistake in the case of human error, and in providing an approximation in the case of inexpressibility. We propose such a method below.

Definition 5. *Let C be a set of causes and Pci be a pci function over C. Let ρ be an ordering of pairs of causes in C. Let Pci_ρ be the sequence of Pci values ordered according to ρ. Let $BPci_\rho$ be the binary string obtained from Pci_ρ by replacing each rif value with 1 and each udm value with 0. Then $BPci_\rho$ is the binary string expression of Pci relative to ordering ρ.*

Definition 6. *Let C be a set of causes, and Pci and Pci' be two pci functions over C. Let $BPci_\rho$ and $BPci'_\rho$ be the binary string expressions of Pci and Pci', relative to ordering ρ of causes in C. Then the following sum is the distance between Pci and Pci':*

$$Dist(Pci, Pci') = \sum_{i=1}^{n(n-1)/2} |BPci_\rho[i] - BPci'_\rho[i]|,$$

where $BPci_\rho[i]$ refers to the ith bit of $BPci_\rho$.

Definition 7. *Let Pci be a pci function over C. The minimum distance of Pci from feasible pci functions over C is*

$$MinDist(Pci) = \min_T Dist(Pci, Pci^T),$$

where Pci^T is the pci function defined by an NIN-AND tree T.

Note $MinDist(Pci) = 0$ if Pci is feasible. Otherwise, we have $MinDist(Pci) > 0$, and there exists a set of minimal NIN-AND trees whose *pci* functions differ from Pci by distance $MinDist(Pci)$. When an expert-specified Pci is detected as infeasible, we propose to return in response either this set of minimal NIN-AND trees or the pairwise interactions implied by them with those interactions that differ from Pci highlighted. This response should be useful for interactive error correction by expert and for choosing the best approximate NIN-AND tree.

For example, suppose that expert specified the *pci* function in Table 1 with an error $pci(c_2, c_4) = rif$. The resultant function Pci is infeasible. It can be shown that $MinDist(Pci) = 1$ and there are six minimal NIN-AND trees whose *pci* functions differ from Pci by distance 1. One of them is Fig. 2.

We exhaustively tested all *pci* functions for $4 \leq n \leq 7$ (Test for $n = 7$ took about 50 hours on a 8-core workstation). For each n value, we define the following *maximal minimum distance* between individual *pci* function and feasible *pci* functions over n causes:

$$MaxMinDist(n) = \max_{Pci} MinDist(Pci),$$

where Pci is a *pci* function over n causes. Table 3 shows this distance for $4 \leq n \leq 7$.

This result shows that for $n = 7$, if a function Pci is infeasible, a minimal NIN-AND tree exists whose *pci* function differs from Pci by no more than four pairwise causal interactions.

Table 3: Maximal minimum distance between individual *pci* function and feasible ones

n	4	5	6	7
$MaxMinDist(n)$	1	2	2	4

Our experiments also show that for many infeasible *pci* functions, a minimal NIN-AND tree exists whose *pci* function differs from the infeasible *pci* function by one pairwise interaction.

For an infeasible *pci* function Pci, there exist generally multiple minimal NIN-AND trees whose *pci* functions differ from Pci by distance $MinDist(Pci)$. We define the *maximum number of minimum distance trees* for *pci* functions over n causes as follows:

$$MaxNumMinDistTree(n) = \max_{Pci} |MinDistTree(Pci)|,$$

where Pci is a *pci* function over n causes, and $MinDistTree(Pci)$ is the set of minimal NIN-AND trees whose *pci* functions differ from Pci by distance $MinDist(Pci)$. Table 3 shows this number for $4 \leq n \leq 7$.

Table 4: Maximum number of minimum distance trees for *pci* functions over n causes

n	4	5	6	7
$MaxNumMinDistTree(n)$	6	10	15	49

Our experiments also show that for many infeasible *pci* functions, $|MinDistTree(Pci)|$ is a small number (less than 10).

These results suggest that acquisition of minimal NIN-AND trees through elicitation of *pci* function is feasible.

6 Conclusion

We presented an algorithm that improves the efficiency for computing conditional probability from an NIN-AND tree model. We proposed another algorithm that computes CPT from a full minimal NIN-AND tree model without having to generate an exponential number of models. Finally, we presented a technique that allows interactive acquisition of NIN-AND trees from pairwise causal interactions.

Future work includes suitability of NIN-AND trees as approximations of arbitrary CPTs, their exploitation in inference, human acquisition testing, and acquisition from learning.

Acknowledgement

Financial support from NSERC Discovery Grant is acknowledged. I thank anonymous reviewers for their comments.

References

S.F. Galan and F.J. Diez. 2000. Modeling dynamic causal interaction with Bayesian networks: temporal noisy gates. In *Proc. 2nd Inter. Workshop on Causal Networks*, pages 1–5.

D. Heckerman and J.S. Breese. 1996. Causal independence for probabilistic assessment and inference using Bayesian networks. *IEEE Trans. on System, Man and Cybernetics*, 26(6):826–831.

J.F. Lemmer and D.E. Gossink. 2004. Recursive noisy OR - a rule for estimating complex probabilistic interactions. *IEEE Trans. on System, Man and Cybernetics, Part B*, 34(6):2252–2261.

P.P. Maaskant and M.J. Druzdzel. 2008. An independence of causal interactions model for opposing influences. In M. Jaeger and T.D. Nielsen, editors, *Proc. 4th European Workshop on Probabilistic Graphical Models*, pages 185–192, Hirtshals, Denmark.

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

Y. Xiang and N. Jia. 2006. Modeling causal reinforcement and undermining with noisy-and trees. In L. Lamontagne and M. Marchand, editors, *Advances in Artificial Intelligence, LNAI 4013*, pages 171–182. Springer-Verlag.

Y. Xiang and N. Jia. 2007. Modeling causal reinforcement and undermining for efficient cpt elicitation. *IEEE Trans. Knowledge and Data Engineering*, 19(12):1708–1718.

Y. Xiang, Y. Li, and J. Zhu. 2009a. Towards effective elicitation of NIN-AND tree causal models. In L. Godo and A. Pugliese, editors, *Inter. Conf. on Scalable Uncertainty Management (SUM 2009), LNAI 5785*, pages 282–296. Springer-Verlag Berlin Heidelberg.

Y. Xiang, J. Zhu, and Y. Li. 2009b. Enumerating unlabeled and root labeled trees for causal model acquisition. In Y. Gao and N. Japkowicz, editors, *Advances in Artificial Intelligence, LNAI 5549*, pages 158–170. Springer.

Y. Xiang. 2010. Generalized non-impeding noisy-AND trees. In *Proc. 23th Inter. Florida Artificial Intelligence Research Society Conf.*, pages 555–560.

Author Index