



Proceedings of the 6th European
Workshop on

Probabilistic Graphical Models

Granada (Spain), 19 - 21 September, 2012

Edited by Andrés Cano, Manuel Gómez-Olmedo and Thomas D. Nielsen



Andrés Cano
Manuel Gómez-Olmedo
Thomas D. Nielsen
(editors)



Proceedings of the Sixth European Workshop
on Probabilistic Graphical Models, PGM'12

Granada, Spain
September 19-21, 2012

Published by: DECSAI, University of Granada
Printed by: Copicentro Granada S.L.
©2012 The authors
ISBN: 978-84-15536-57-4
Cover design by M. Gómez-Olmedo
This book has been typeset using L^AT_EX
Printed in Spain
Depósito Legal: GR-2569/2012

Contents

Preface	vii
Organisation	ix
Conference Papers	
Active learning by the naive credal classifier	
Alessandro Antonucci, Giorgio Corani, and Sandra Gabaglio	3
ProbModelXML. A format for encoding probabilistic graphical models	
Manuel Arias, Francisco Javier Díez, Miguel Palacios-Alonso, and Íñigo Bermejo	11
Gaussian join tree classifiers with applications to mass spectra classification	
Victor Bellón, Jesús Cerquides, and Ivo Grosse	19
Interactive learning of Bayesian networks using OpenMarkov	
Íñigo Bermejo, Jesús Oliva, Francisco Javier Díez, and Manuel Arias	27
Tools and algorithms for causally interpreting directed edges in maximal ancestral graphs	
Giorgos Borboudakis, Sofía Triantafillou, and Ioannis Tsamardinos	35
Approximate inference in influence diagrams using binary trees	
Rafael Cabañas de Paz, Manuel Gómez-Olmedo, and Andrés Cano	43
The same-decision probability: A new tool for decision making	
Suming Chen, Arthur Choi, and Adnan Darwiche	51
Piecewise linear approximations of nonlinear deterministic conditionals in continuous Bayesian networks	
Barry R. Cobb and Prakash P. Shenoy	59
Approximating the distribution of a sum of log-normal random variables	
Barry R. Cobb, Rafael Rumí, and Antonio Salmerón	67
A Bayesian network model for predicting the outcome of in vitro fertilization	
Giorgio Corani, Cristina Magli, Alessandro Giusti, Luca Gianaroli, and Luca Gambardella	75
Decision analysis networks	
Francisco Javier Díez, Manuel Luque, and Caroline Leonore König	83

Gibbs sampling for parsimonious Markov models with latent variables	
Ralf Eggeling, Pierre-Yves Bourguignon, André Gohr, and Ivo Grosse	91
A framework for development, teaching and deployment of inference algorithms	
Sander Evers and Peter J.F. Lucas	99
Meta-prediction of semi-naive Bayesian network classifiers based on dataset complexity characterization	
M. Julia Flores, José A. Gámez, and Ana M. Martínez	107
Gaussian process regression with censored data using expectation propagation	
Perry Groot and Peter J.F. Lucas	115
Two optimal strategies for active learning of causal models from interventions	
Alain Hauser and Peter Bühlmann	123
Probabilistic reasoning with temporal indeterminacy	
Maarten van der Heijden and Peter J.F. Lucas	131
Hybrid binary-chain multi-label classifiers	
Pablo Hernandez-Leal, Felipe Orihuela-Espina, L. Enrique Sucar, and Eduardo F. Morales	139
Estimation of effect size posterior using model averaging over Bayesian network structures and parameters	
Gabor Hullam and Peter Antal	147
Latent tree copulas	
Sergey Kirshner	155
Learning mixtures of truncated basis functions from data	
Helge Langseth, Thomas Dhyre Nielsen, and Antonio Salmerón	163
Inference in hybrid Bayesian networks with mixtures of truncated basis functions	
Helge Langseth, Thomas Dhyre Nielsen, Rafael Rumí, and Antonio Salmerón	171
Qualitative chain graphs and their use in medicine	
Martijn Lappenschaar, Arjen Hommersom, and Peter J.F. Lucas	179
A depth-first search algorithm for optimal triangulation of Bayesian network	
Chao Li and Maomi Ueno	187
Decision-theoretic troubleshooting: hardness of approximation	
Václav Lín	195
A novel LTM-based method for multi-partition clustering	
Tengfei Liu, Nevin L. Zhang, Kin Man Poon, Hua Liu, and Yi Wang	203
Learning mixtures of polynomials from data using B-spline interpolation	
Pedro L. López-Cruz, Concha Bielza, and Pedro Larrañaga	211

A comparison of popular fertility awareness methods to a DBN model of the woman's monthly cycle	219
Anna Lopińska-Dubicka and Marek J. Drużdżel	
On the importance of elimination heuristics in lazy propagation	227
Anders L. Madsen and Cory J. Butz	
A bounded error, anytime, parallel algorithm for exact Bayesian network structure learning	235
Brandon Malone and Changhe Yuan	
An interactive approach for cleaning noisy observations in Bayesian networks with the help of an expert	243
Andrés R. Masegosa and Serafín Moral	
Learning AMP chain graphs under faithfulness	251
Jose M. Peña	
Prediction of coffee rust disease using Bayesian networks	259
Cora B. Pérez-Ariza, Ann E. Nicholson, and M. Julia Flores	
Generalised co-variation for sensitivity analysis in Bayesian networks	267
Silja Renooij	
Tractable inference in hybrid Bayesian networks with deterministic conditionals using re-approximations	275
Rafael Rumí, Antonio Salmerón, and Prakash P. Shenoy	
Mixtures of bagged Markov tree ensembles	283
François Schnitzler, Pierre Geurts, and Louis Wehenkel	
Graphical inequality diagnostics for phylogenetic trees	291
Nathaniel Shiers and Jim Q. Smith	
Learning multivariate regression chain graphs under faithfulness	299
Dag Sonntag and Jose M. Peña	
Integer linear programming approach to learning Bayesian network structure: towards the essential graph	307
Milan Studeny	
The Bayesian Chow-Liu algorithm	315
Joe Suzuki	
Learning high-dimensional mixed graphical models with missing values	323
Inma Tur and Robert Castelo	
Non-informative Dirichlet score for learning Bayesian networks	331
Maomi Ueno and Masaki Uto	
New local move operators for Bayesian network structure learning	339
Jimmy Vandel, Brigitte Mangin, and Simon de Givry	

Flood damage and influencing factors: A Bayesian network perspective	
Kristin Vogel, Carsten Riggelsen, Bruno Merz, Heidi Kreibich, and Frank Scherbaum	347
Computationally efficient probabilistic inference with noisy threshold models based on a CP tensor decomposition	
Jiří Vomlel and Petr Tichavský	355
Bayesian network inference with NIN-AND tree models	
Yang Xiang	363
Indexes	371
Subject Index	371
Author Index	373

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. It provides a forum for discussion of the latest research developments in these fields. The workshop is organized so as to facilitate discussions and collaboration among the participants also outside the workshop sessions.

PGM 2012 is the sixth workshop in the series, and will take place in Granada, Spain, in September 19–21, 2012. Previous meetings were held in Cuenca, Spain (PGM 2002), Leiden, Netherlands (PGM 2004), Prague, Czech Republic (PGM 2006), Hirtshals, Denmark (PGM 2008) and, Helsinki, Finland (PGM 2010).

This year there were 63 papers submitted to the workshop (a record with respect to the previous editions of the workshop) with authors coming from 23 different countries. The reviewing process took place in the period from June 12th to July 18th, 2012. The papers went through a rigorous reviewing process, where the majority of the papers were reviewed by three members of the PGM 2012 programme committee and with each PC member reviewing around four papers. The reviews were handled electronically using the EasyChair conference system.

Of the 63 submissions, 47 were accepted for presentation at the workshop (although the authors of one of the accepted papers had to withdraw their paper from the workshop). Each accepted paper is given a slot in both a plenary and a poster session taking place on the same day. In addition to the presentation of the technical papers, we are very pleased to have three distinguished invited speakers: Norman Fenton (Queen Mary University of London) who will give a talk on *Improving Legal Reasoning with Bayesian Networks* as well as Pedro Larrañaga and Concha Bielza (Technical University of Madrid) who will give a presentation about *Bayesian Networks in Neuroscience*.

We would like to give our sincere thanks to the 56 members of the programme committee for their outstanding work during the reviewing process, which is of course crucial for a successful workshop. In addition to the programme committee members, there were also 9 additional reviewers to whom we are also most grateful. The reviews that we received were of general high quality and included constructive comments to help the authors improve upon their papers.

We would also like to thank the members of the Local Organizing Committee, which helped us with a multitude of issues. Special thanks to Serafín Moral for his help in the production of the proceedings.

Further thanks are devoted to the sponsors, who have provided infrastructure and financial support: University of Granada, High Technical School of Computer Science and Telecommunication from the University of Granada, Department of Computer Science and Artificial Intelligence (University of Granada), the Research Centre on Computer Science and Communication Thecnologies (University of Granada), Ministry of Economy and Competitivility (Spain) under project TIN2011-15936-E, Hugin Expert A/S, and Dezide.

Our wishes is for the participants of the PGM 2012 workshop to appreciate both the beauty of Granada and the scientific program of the workshop. We also hope and expect that people will enjoy the taste of tapas in Granada.

We hope that these proceedings will help convey the state of the art of probabilistic graphical models, raise interest, and contribute to the further dissemination and discussion of the fascinating ideas of this active and highly relevant research field.

Andrés Cano, Manuel Gómez-Olmedo, and Thomas Dyrre Nielsen

September 2012

Organisation

Program Committee Board

Andrés Cano, Granada University, Spain
Manuel Gómez-Olmedo, Granada University, Spain
Thomas D. Nielsen, Aalborg University, Denmark

Program Committee Members

Alessandro Antonucci, Switzerland	Petri Myllymäki, Finland
Concha Bielza, Spain	Jens Nielsen, UK
Wray Buntine, Australia	Kristian Olesen, Denmark
Cory J. Butz, Canada	Thorsten Ottosen, Denmark
Adnan Darwiche, USA	José M. Peña, Sweden
Luis M. de Campos, Spain	José M. Puerta, Spain
Francisco J. Díez, Spain	Silja Renooij, The Netherlands
Marek J. Druzdzel, USA , Poland	Carsten Riggelsen, Germany
Ad Feelders, The Netherlands	David Ríos, Spain
M. Julia Flores, Spain	Teemu Roos, Finland
José A. Gámez, Spain	Antonio Salmerón, Spain
Christophe Gonzales, France	Prakash P. Shenoy, USA
Patrik Hoyer, Finland	Jim Q. Smith, UK
Juan F. Huete, Spain	Harald Steck, USA
Manfred Jaeger, Denmark	Fabio Stella, Italy
Finn V. Jensen, Denmark	Milan Studený, Czech Republic
Radim Jiroušek, Czech Republic	Enrique Súcar, México
Kristian Kersting, Germany	Rosario Susi, Spain
Mikko Koivisto, Finland	Marco Valtorta, USA
Petri Kontkanen, Finland	Linda van der Gaag, The Netherlands
Helge Langseth, Norway	Paolo Viappiani, Denmark
Pedro Larrañaga, Spain	Jirka Vomlel, Czech Republic
Philippe Leray, France	Marta Vomlelová, Czech Republic
José A. Lozano, Spain	Jon Williamson, UK
Peter J.F. Lucas, The Netherlands	Pierre-Henri Wuillemin, France
Paloma Main, Spain	Yang Xiang, Canada
Stijn Meganck, Belgium	Marco Zaffalon, Switzerland
Serafín Moral, Spain	Nevin L. Zhang, Hong Kong

Additional Reviewers

Hans Bodlaender
Nikolaos Gianniotis
Arjen Hommersom
Václav Lín
Tengfei Liu

Luis de la Ossa
Pekka Parviainen
Mohamed Sharaf
Lucie Vachova

Organizing Committee Supervisors

Andrés Cano, Granada University, Spain
Manuel Gómez-Olmedo, Granada University, Spain

Organizing Committee Members

Joaquín Abellán, Granada University, Spain
Silvia Acid, Granada University, Spain
Rafael Cabañas, Granada University, Spain
Luis M. de Campos, Granada University, Spain
Juan M. Fernández-Luna, Granada University, Spain
Francisco J. García-Castellano, Granada University, Spain
Juan Huete, Granada University, Spain
Andrés Ramón Masegosa, Granada University, Spain
Serafin Moral, Granada University, Spain
Cora Beatriz Pérez-Ariza, Granada University, Spain

Sponsors



DECSAI
Departamento de Ciencias
de la Computación e I.A.
Universidad de Granada



ugr | Universidad
de **Granada**



Escuela Técnica Superior
de Ingenierías Informática
y de Telecomunicación
ETSIIT



HUGINEXPERT
The leading decision support tool

DÉZIDE

Proyecto TIN2011-15936-E



Conference Papers

Active Learning by the Naive Credal Classifier

Alessandro Antonucci
IDSIA, Switzerland
alessandro@idsia.ch

Giorgio Corani
IDSIA, Switzerland
giorgio@idsia.ch

Sandra Gabaglio
SUPSI, Switzerland
sandra.gabaglio@supsi.ch

Abstract

In standard classification a training set of supervised instances is given. In a more general setup, some supervised instances are available, while further ones should be chosen from an unsupervised set and then annotated. As the annotation step is costly, *active learning* algorithms are used to select which instances to annotate to maximally increase the classification performance while annotating only a limited number of them. Several active learning algorithms are based on the naive Bayes classifier. We work instead with the naive *credal* classifier, namely an extension of naive Bayes to imprecise probability. We propose two novel methods for active learning based on the naive credal classifier. Empirical comparisons show performance comparable or slightly superior to that of approaches solely based on the naive Bayes.

1 Introduction

In standard classification problems the goal is to learn a classifier able to assign class labels to the unsupervised instances of a *test set*, given a *training set* of supervised instances. If we assume the class variable directly unobservable, the training should be regarded as the output of an annotation process over a set of unsupervised instances.

Active learning (AL) (Settles, 2000) is the process of selecting the instances to be annotated, among all the unsupervised ones. AL algorithms rank the unsupervised instances by means of a *score*; the instances with highest score are then annotated. The classification accuracy increases with the amount of supervised instances, because the variance component of the classification error decreases with the size of the training set. The goal of AL is thus to

select the most meaningful instances to be annotated, in order to maximally increase the classifier performance. Several works on active learning have used naive Bayes (NBC) as a classifier; see for instance (Chai et al., 2004; McCallum and Nigam, 1998).

NBC has been extended to cope with sets of probabilities by the so-called *naive credal classifier* (NCC, Corani and Zaffalon (2008)). Instead of a single Dirichlet prior, the parameters are estimated on the basis of a set of priors modeling a condition of near-ignorance *a priori* to achieve more robust and reliable estimates. NCC automatically detects *prior-dependent* instances, namely instances whose most probable class changes when different priors are considered. On these instances, the set of posterior distributions for the class given the attributes is particularly uncertain (i.e., large) and NCC returns multiple classes.

We present two measures of uncertainty based on NCC, which estimate how strong are the dominances among the classes (see Sect. 5.1) and how large the set of posterior distributions for the class is (see Sect. 5.2). In Sect. 7 we compare the AL results obtained with these new methods with their Bayesian counterparts. Moreover, following the ideas in (McCallum and Nigam, 1998), in Sect. 6 we also implement a density-weighted approach; density-weighted approaches are designed to avoid annotating instances which are controversial but very rare, and thus have little impact on the average accuracy of the classifier. We thus also present a density-weighted approach based on NCC.

2 Naive Bayes and Credal Classifiers

We consider a classification task with class variable C , taking values in a finite set \mathcal{C} , based on a set of *attributes* $\mathbf{A} := (A_1, \dots, A_k)$, where for each $i = 1, \dots, k$, A_i takes values in the finite set \mathcal{A}_i . We denote by $|\mathcal{C}|$ the cardinality of \mathcal{C} , i.e., the number of possible classes, and similarly for the attributes. A training set of joint observations for these variables is available, i.e., $\mathcal{D} := \{(c^{(i)}, \mathbf{a}^{(i)})\}_{i=1}^d$. Learning a *classifier* from data \mathcal{D} means to implement a map $\times_{i=1}^k \mathcal{A}_i \rightarrow \mathcal{C}$ assigning a class label to any joint observation of the attributes. In particular, probabilistic classifiers are obtained by learning from \mathcal{D} a joint distribution $P_{\mathcal{D}}(C, \mathbf{A})$. Given this distribution, the class label assigned to a (joint) test instance of the attributes, say $\mathbf{a} \in \times_{i=1}^k \mathcal{A}_i$ is:¹

$$c^* := \arg \max_{c \in \mathcal{C}} P_{\mathcal{D}}(c, \mathbf{a}). \quad (1)$$

The *naive Bayes classifier* (NBC) is a probabilistic classifier based on the assumption of conditional independence between the attributes given the class variables, this inducing the factorization $P(c, \mathbf{a}) = P(c) \prod_{i=1}^k P(a_i|c)$, where a_i is the value of A_i consistent with \mathbf{a} .

NBC, whose parameters are estimated through a standard Bayesian approach, has been extended to imprecise probability by

¹In both (1) and (2) the joint probability can replace the corresponding conditional for the class because of the proportionality relation among them.

the so-called *naive credal classifier* (NCC, Corani and Zaffalon (2008)). While NBC learns from data a joint distribution $P_{\mathcal{D}}$, NCC learns a joint *credal set*, i.e., a set of joint distributions $\mathcal{P}_{\mathcal{D}}(C, \mathbf{A}) := \{P_{\theta}(C, \mathbf{A})\}_{\theta \in \Theta}$, where θ is a parameter indexing a family of NBC specifications. NCC does not simply return the class which is the most probable a posteriori as in (1), since multiple posterior distributions now characterize the model; it instead returns the *non-dominated* classes. A class c' *dominates* an alternative class c'' if c' is more probable than c'' for each distribution in the joint credal set; more formally, dominance holds iff:

$$\gamma_{\mathbf{a}}(c', c'') := \inf_{\theta \in \Theta} \frac{P_{\theta}(c', \mathbf{a})}{P_{\theta}(c'', \mathbf{a})} > 1. \quad (2)$$

Such a dominance test can be efficiently evaluated by the algorithm described in Corani and Zaffalon (2008). NCC compares the classes in a pairwise fashion to identify the set of non-dominated classes $\mathcal{C}^* \subseteq \mathcal{C}$. Corani and Zaffalon (2008) have shown by extensive experiments that instances for which \mathcal{C}^* contains more than a single class are often misclassified by NBC.

3 Active Learning (AL)

Let us focus on the way the training set can be obtained. Assuming the class directly unobservable, \mathcal{D} should be regarded as the result of the *annotation*² of a set of unsupervised observations of the attributes. In general situations, only some of the instances might be annotated, this requiring the choice of which instances to annotate. An *active learning* (AL) algorithm provides a strategy for identifying the instances to be annotated. In particular, we consider the situation where a supervised training set is already available and l instances should be picked from an unsupervised set, called the *active learning set*. The performance indicator of an AL algorithm is the growth of the accuracy of the classifier when the l instances are annotated

²We call annotation the process of assigning the proper class label to an unsupervised instance of the attributes. This is achieved by an *annotation oracle* corresponding to a perfect (i.e., 100% accuracy) classifier.

and added to the training set. Even a random picking of the instances generally increases the accuracy; thus an AL algorithm should produce a quicker increase of accuracy over selecting the instances in a random fashion.

4 Active Learning with Naive Bayes

Let us denote by \mathcal{D}_A the *active learning set*, namely the set of instances from which to select the instances to be annotated. We consider a setting in which at each iteration the AL algorithm selects from \mathcal{D}_A the subset of l instances with the highest *score*. In the following we describe two well-known AL algorithms, which can be used with NBC.

4.1 Uncertainty Sampling (US)

One of the simplest and most commonly used query framework is uncertainty sampling: the active learner queries the instances whose labels are the most uncertain (Settles, 2000, pag. 12). The score is defined as follows:

$$\text{score}(\mathbf{a}) := -P(c^*|\mathbf{a}), \quad (3)$$

where c^* is the most probable class as in (1). Because of the minus, the most uncertain instances will have the highest score. For binary classification, uncertainty sampling queries the instance whose posterior probability of being positive is nearest to 0.5.

4.2 Query by Committee (QbC)

Following the definition of (Settles, 2000, pag. 15), “*the QbC approach involves maintaining a committee of models which are all trained on the current labeled set, but represent competing hypotheses. Each committee member is then allowed to vote on the labelings of query candidates. The most informative query is considered to be the instance about which they most disagree.*” We implement QbC by generating q different bootstrap replicates $\{\mathcal{D}_j\}_{j=1}^q$ of the training set and then learning a different NBC from each of them. Denote by $P^{(j)}(C|\mathbf{a})$ the posterior distributions computed by the j -th NBC;

the center of mass of such posteriors is:

$$\tilde{P}_{\mathbf{a}}(c) := \frac{1}{q} \sum_{j=1}^q P^{(j)}(c|\mathbf{a}). \quad (4)$$

The score proposed in (McCallum and Nigam, 1998) is the average of the KL divergence between the members of the committee and the center of mass, i.e.,³

$$\text{score}(\mathbf{a}) := \frac{1}{q} \sum_{j=1}^q \text{KL}[P^{(j)}(C|\mathbf{a}), \tilde{P}_{\mathbf{a}}(C)]. \quad (5)$$

The more the posterior distributions of the committee members disagree, the higher the score.

5 NCC-based Active Learning

As pointed out in Sect. 2, NCC natively identifies prior-dependent instances, for which it returns multiple class labels in output. The more the returned classes, the harder the instance to be classified, as shown in Corani and Zaffalon (2008). However the number of non-dominated classes is not a viable score for active learning: such an approach would not be able to discriminate among instances which have received the same number of class labels. Further refinements are thus necessary for a NCC-based AL approach; we present two alternative approaches in the following.

5.1 Credal Uncertainty Sampling

Let us start by considering a binary class, i.e., $\mathcal{C} = \{c', c''\}$. The instances for which the dominance is more clear are characterized by higher values of the maximum ratio between the posterior probabilities of the two classes, introduced in (2). We thus propose the following AL score:

$$\text{score}(\mathbf{a}) := -\max[\gamma_{\mathbf{a}}(c', c''), \gamma_{\mathbf{a}}(c'', c')]. \quad (6)$$

where $\gamma_{\mathbf{a}}(c', c'')$ has been defined in (2). The more negative this score, the clearer the superiority of a class over another one; consistently with the previous ones, higher scores correspond to instances deemed to be more uncertain.

³The Kullback-Leibler divergence with discrete variables is $\text{KL}[P', P''] := -\sum_{c \in \mathcal{C}} P'(c) \cdot \log \frac{P''(c)}{P'(c)}$.

In particular with scores smaller than -1 , one of the two classes dominates the other as in (2). If the score is greater than -1 , there is no dominance among the two classes. Thus, any instance for which NCC has returned two classes should be regarded as more uncertain than any instance for which NCC has returned a single class. For data sets with more than two classes, the score is generalized as follows:

$$-\sum_{c' \in \mathcal{C}} \sum_{c'' \in \mathcal{C} \setminus \{c'\}} \max[\gamma_{\mathbf{a}}(c', c''), \gamma_{\mathbf{a}}(c'', c')]. \quad (7)$$

We call this approach *credal uncertainty sampling* (CUS); it tries to identify how strong is the dominance among the classes, and thus it can be seen as a credal counterpart of the uncertainty-sampling based on NBC.

5.2 Credal Query by Committee

QbC artificially generates multiple NBCs through a bootstrap resampling of the dataset. Notice that NCC constitutes a committee of NBCs as well, although different from QbC. Each committee member of NCC is in fact induced by the updating of a different prior (those in the *imprecise Dirichlet model*, see App. A); therefore, each member computes a different posterior. Yet, the set $\mathcal{P}(C|\mathbf{a}) := \{P_\theta(C|\mathbf{a})\}_{\theta \in \Theta}$ has an infinite number of elements, this preventing a straightforward application of the ideas in Sect. 4.2. Nevertheless, the set of posteriors is convex and its *extreme* points, namely those which cannot be obtained as a convex combination of other posteriors, are only a finite number.⁴ We call *Credal Query by Committee* (CQbC) a QbC-like approach which adopts as committee members only the extremes of the NCC set of posterior distribution.

To support the idea of removing non-extreme members from the committee, it could be worth noticing that computing the bounds of an expectation with respect to a set of distributions is a LP task, whose optimum lies on an extreme point of the feasible region.

To obtain the extremes of $\mathcal{P}(C|\mathbf{a})$, we first

evaluate its lower and upper bounds, i.e.,

$$\underline{P}(c|\mathbf{a}) := \inf_{\theta \in \Theta} P_\theta(c|\mathbf{a}), \quad (8)$$

$$\overline{P}(c|\mathbf{a}) := \sup_{\theta \in \Theta} P_\theta(c|\mathbf{a}). \quad (9)$$

The values and the formulae to compute these bounds are in App. A. Then we consider the set of distributions consistent with these bounds, i.e., set $\mathcal{P}'(C|\mathbf{a})$ defined as:

$$\left\{ P(C) \mid \begin{array}{l} \underline{P}(c|\mathbf{a}) \leq p(c) \leq \overline{P}(c|\mathbf{a}) \forall c \in \mathcal{C} \\ \sum_{c \in \mathcal{C}} P(c) = 1 \end{array} \right\}. \quad (10)$$

The extremes of $\mathcal{P}'(C|\mathbf{a})$ can be obtained from the constraints in (10) by the fast algorithm in (de Campos et al., 1994), their number being bounded by the factorial of $|\mathcal{C}|$.

Although, in general, $\mathcal{P}'(C|\mathbf{a})$ is only an outer approximation of $\mathcal{P}(C|\mathbf{a})$, the two credal sets are known to be very close (e.g., see the discussion in Antonucci and Cuzzolin (2010)). We can therefore evaluate the score in (5) with the elements $\{P^{(j)}(C|\mathbf{a})\}_{j=1}^q$ coinciding with the extremes of $\mathcal{P}'(C|\mathbf{a})$. This QbC-like approach will be called *credal query by committee* (CQbC).

In Sect. 7 we show that also in the traditional QbC what matters are in fact the extreme members, namely those whose opinion cannot be obtained as a convex combination of other members: removing members of the committee which are in the convex hull of the others basically does not affect the QbC ranks.

6 Density-weighted approaches

For a further improvement in the AL performance, (McCallum and Nigam, 1998) proposed to weight the score of each instance on the basis of its representativeness. In the words of (Settles, 2000, pag. 25) “*the main idea is that informative instances should not only be those which are uncertain, but also those which are representative of the underlying distribution (i.e., inhabit dense regions of the input space)*”. In particular a similarity measure among the attributes instances is defined and then summed over all the instances. As an alternative to this information-theoretic approach, here we propose a purely probabilistic approach, where the

⁴See the *lower envelope theorem* in (Walley, 1991).

level of representativeness of each instance corresponds to the marginal probability of the joint observation of the instances, i.e., we rescale the NBC-based scores as follow:⁵

$$\text{score}'(\mathbf{a}) = \text{score}(\mathbf{a}) \cdot P(\mathbf{a}). \quad (11)$$

For the NBC, the weights are simply:

$$P(\mathbf{a}) = \sum_{c \in \mathcal{C}} \left[P(c) \prod_{i=1}^k P(a_i|c) \right]. \quad (12)$$

This idea can be easily extended to the NCC by simply replacing in (12) the probability of the joint observation of the attributes with the corresponding *lower* probability, i.e.,

$$\underline{P}(\mathbf{a}) = \inf_{\theta \in \Theta} P_\theta(\mathbf{a}). \quad (13)$$

A formula for this bound is in App. B.

7 Experiments

We empirically compare the AL algorithms based on NCC with their counterparts based on NBC on different data sets from the UCI repository. Notice that we use the AL algorithms based on NCC to rank the instances and to select which ones should be annotated; then, we use the annotated instances to update the parameters of a standard NBC classifier.

For instance, to compare uncertainty sampling (US) and credal uncertainty sampling (CUS), we proceed as follows: we randomly draw a training set of $n_0=10$ instances and a test set of 100 instances; we generate both training and test set in a stratified way, namely they contain the same proportion of instances from the various classes as the original data set. We then estimate the parameters of NBC from the training set; this starting classifier is denoted as $\text{NBC}^{(0)}$. Then, we rank the unsupervised instances in \mathcal{A} according to *uncertainty sampling*, using $\text{NBC}^{(0)}$; we then annotate the $l=5$ instances with highest score and use them to revise $\text{NBC}^{(0)}$, thus obtaining $\text{NBC}_{\text{US}}^{(1)}$, which

⁵We assume that the scores are nonnegative values. If this is not the case we can always sum to all of them the minimum score to obtain nonnegative values.

denotes a NBC actively learned by uncertainty sampling at the first iteration. We then update the active learning set as $\mathcal{D}_{\text{US}}^{(1)} := \mathcal{D}_A \setminus \overline{\mathcal{D}}_{\text{US}}^{(1)}$, where $\overline{\mathcal{D}}_{\text{US}}^{(1)}$ denotes the l instances selected by uncertainty sampling for the first update; we then evaluate the accuracy of $\text{NBC}_{\text{US}}^{(1)}$ on the test set. This procedure (scoring instances, updating classifier and active learning set, assessing accuracy on the test set) is iterated until the active learning set is empty.

To assess *credal uncertainty sampling* we follow the very same procedure, beginning from the same starting point, namely $\text{NBC}^{(0)}$, with the only difference of using the score in (6) to rank the instances to be annotated. Namely, we use the $l=5$ instances with highest CUS score to update the NBC parameters, yielding $\text{NBC}_{\text{CUS}}^{(1)}$, which denotes a NBC actively learned by *credal* uncertainty sampling, after the first active learning iteration. We then update the active learning set as $\mathcal{D}_{\text{CUS}}^{(1)} = \mathcal{D}_A \setminus \overline{\mathcal{D}}_{\text{CUS}}^{(1)}$. The accuracy of $\text{NBC}_{\text{CUS}}^{(1)}$ is then assessed on the test set, and so on. We remark that the starting training set and classifier $\text{NBC}^{(0)}$ is identical for the different AL algorithms, thus allowing to fairly compare them. We repeat the procedure 50 times for each data set and we report the average results.

We compare the following pairs of methods: US versus CUS; QbC versus CQbC. To choose the number of member of the QbC committee, we first noticed that: “*there is no general agreement in the literature on the appropriate committee size to use, which may in fact vary by model class or application. However, even small committee sizes (e.g., two or three) have been shown to work well in practice*” (Settles, 2000, pag. 16). After some preliminary experiments, $q=7$ seemed a viable choice for this parameter.

It is hard to summarize in a single indicator the performance of an AL algorithm; it is instead much more meaningful showing the whole trajectory of accuracy for training sets of growing dimensions, as shown in Fig. 1. Although we analyzed 16 data sets, for the lack of space we graphically present only part of the results, selecting some representative examples.

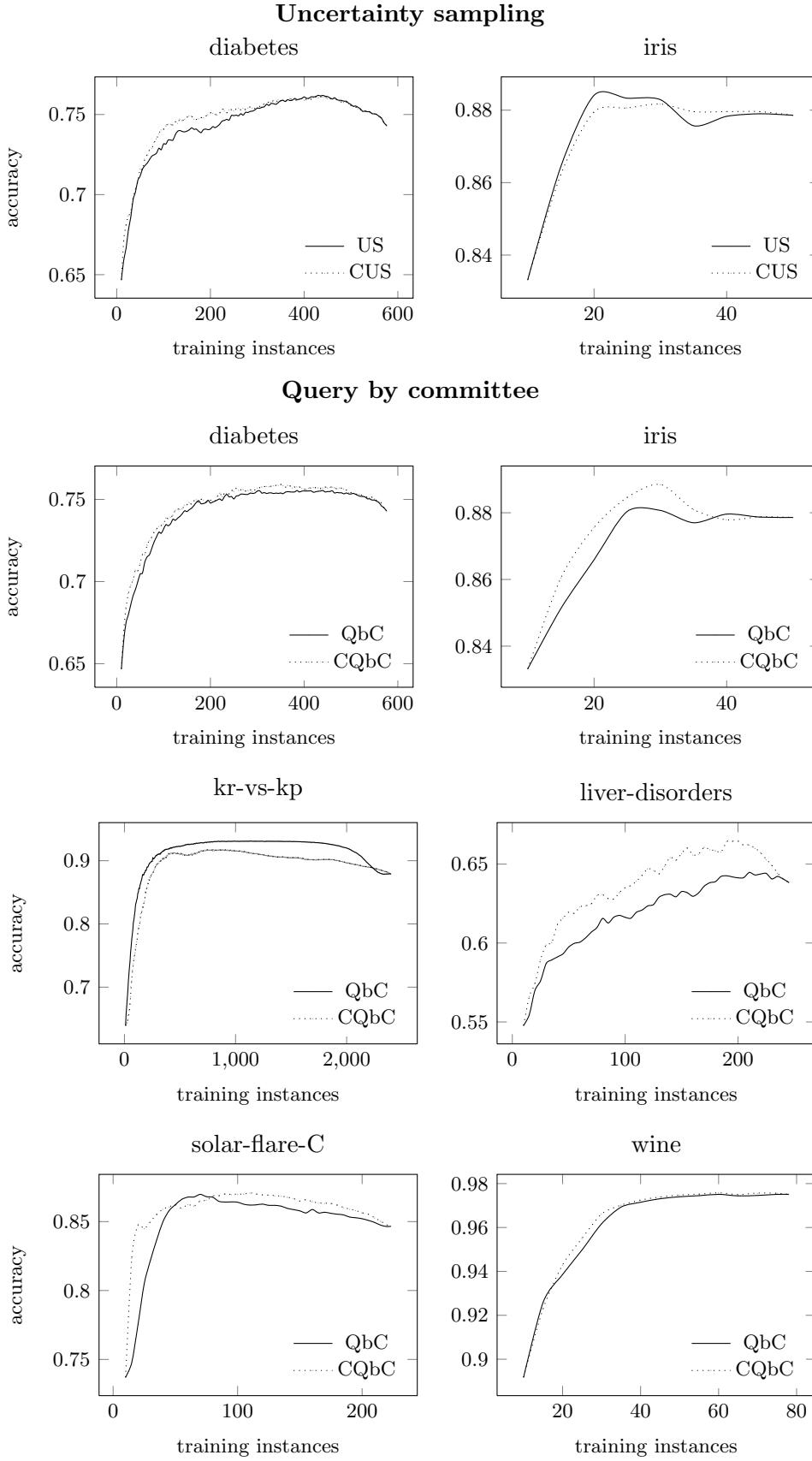


Figure 1: Experimental results on different data sets.

The comparison between US and CUS did not show a clear winner among the two approaches. Most commonly, the accuracy trajectories of the two methods cross different times in a single graph; we conclude that the two approaches perform in a substantially equivalent way. Some examples are shown in the top row of Fig. 1.

Notably, when using criteria different from the random sampling, the training set is not necessarily representative of the underlying population: the variance component of the classification error might therefore not decrease when the training set size increases. This explains the local decreases in some trajectories. On the contrary, the accuracy trajectories for the random sampling are monotonically increasing, but being always under the trajectories associated to both US and QbC (Settles, 2000) are not shown in the plots.

The comparison between QbC and CQbC shows instead a certain superiority for CQbC, as can be inferred from Fig. 1; an exception is however found on the kr-vs-kp data set, whose results are shown in the third row of Fig. 1. Overall the results suggest that the committee obtained by updating a convex set of priors outperforms the committee obtained by building a finite number of bootstrap replicates. A possible explanation is that the former approach has an infinite number of members, which might be beneficial for the accuracy of the committee.

An interesting empirical finding is that however in the committee member what really matters are the extreme members. The similarity among two ranks can be assessed by the Spearman correlation, whose maximum value is 1. We found a Spearman correlation consistently higher than 0.9 between the rank computed, on the instances of the active learning set, by QbC and CQbC restricted to its extreme members. In binary classification, the two extreme members of the committee are those which assign the highest and the lowest probability to the positive class; the extreme members change instance by instance, and therefore this finding does not allow to prune the committee. Yet, it provides some empirical support for the CQbC approach, which relies on the upper and lower

probability for the class.

Finally, the probabilistic version of the density-weighted approach proposed in Sect. 6 did not generally provide a significant improvement in the performance. Our explanation for this result is that the computed probability of an instance are negatively affected by the adopted naive architecture, which is known to be a bad estimator of the joint probability. Since weighting the scores by a density measure is generally recognized to boost the AL performance, we see as a future research direction the adoption of more realistic topologies for the computation of the probability of the instance.

8 Conclusions

In this paper we proposed two new active learning algorithms based on the naive credal classifier, rather than on the traditional naive Bayes. Results are especially encouraging for the credal query-by-committee approach; future research direction include the refinement of the density-weighted approach, in order to further boost the active learning performance.

Acknowledgements

The research in this paper has been partially supported by the Swiss NSF grants no. 200020-132252 and by the Hasler foundation grant n. 10030. We also thank the first reviewer for his/her constructive comments.

References

- A. Antonucci and F. Cuzzolin. 2010. Credal sets approximation by lower probabilities: application to credal networks. In *Proceedings of IPMU 2010*, pages 716–725.
- X. Chai, L. Deng, Q. Yang, and C.X. Ling. 2004. Test-cost sensitive naive bayes classification. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 51–58. IEEE.
- G. Corani and M. Zaffalon. 2008. Learning reliable classifiers from small or incomplete data sets: The naive credal classifier 2. *The Journal of Machine Learning Research*, 9:581–621.
- L.M. de Campos, J.F. Huete, and S. Moral. 1994. Probability intervals: a tool for uncertain reason-

ing. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2:167–196.

- A. McCallum and K. Nigam. 1998. Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 350–358. Morgan Kaufmann Publishers Inc.
- B. Settles. 2000. Active learning literature survey. *Computer Sciences Technical Report*, 1648.
- P. Walley. 1991. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, New York.

Appendix A: Lower conditional

Consider the computation of the posterior probability for the class given the attributes. For the NBC we have:

$$P(c|\mathbf{a}) := \frac{P(c, \mathbf{a})}{\sum_{c' \in \mathcal{C}} P(c', \mathbf{a})}, \quad (14)$$

which, by exploiting the NBC factorization, with straightforward algebra, rewrites as:

$$P(c|\mathbf{a}) = \left[1 + \frac{\sum_{c' \neq c} P(c') \prod_{i=1}^k P(a_i|c')}{P(c) \prod_{i=1}^k P(a_i|c)} \right]^{-1}. \quad (15)$$

The NBC probabilities are obtained from the counts in \mathcal{D} as:

$$P(a_i|c) := \frac{n(a_i, c) + st(a_i, c)}{n(c) + s}, \quad (16)$$

$$P(c) := \frac{n(c) + st(c)}{n(\cdot) + s}, \quad (17)$$

this corresponding to a Bayesian learning approach with Dirichlet priors with parameters $\{t(c), t(a_i, c)\}$. Unlike NBC, NCC consider a set of priors, whose parameters are free to vary in the following sets:⁶

$$\left\{ t(C) \left| \begin{array}{l} \frac{\epsilon}{|\mathcal{C}|} \leq t(c) \leq \frac{\epsilon}{|\mathcal{C}|} + (1 - \epsilon) \\ \forall c \in \mathcal{C}, \\ \sum_{c \in \mathcal{C}} t(c) = 1 \end{array} \right. \right\}, \quad (18)$$

$$\left\{ t(A_i|c) \left| \begin{array}{l} \frac{\epsilon}{|\mathcal{A}_i|} \leq t(a_i, c) \leq \frac{\epsilon}{|\mathcal{A}_i|} + (1 - \epsilon) \\ \forall a_i \in \mathcal{A}_i \\ \sum_{a_i \in \mathcal{A}_i} t(a_i, c) = 1 \end{array} \right. \right\}. \quad (19)$$

⁶This is the so-called imprecise Dirichlet model which, after an ϵ regularization to avoid problems with zero counts. In particular, this is called the *local* version of the IDM model.

The problem is therefore minimize (15) with the probabilities defined as in (16) and (17) with respect to the constraints in (18) and (19). By simply exploiting the monotonicity of function $f(x) := \frac{1}{1+x}$ and with simple algebra we obtain:

$$\underline{P}(c|\mathbf{a}) = \sum_{c' \in \mathcal{C} \setminus \{c\}} \frac{n(c') + s \frac{\epsilon}{|\mathcal{C}|}}{n(c) + s[\frac{\epsilon}{|\mathcal{C}|} + (1 - \epsilon)]} \cdot \left(\frac{n(c) + s}{n(c') + s} \right)^k \prod_{i=1}^k \frac{n(a_i, c') + s \frac{\epsilon}{|\mathcal{A}_i|}}{n(a_i, c) + s[\frac{\epsilon}{|\mathcal{A}_i|} + (1 - \epsilon)]}. \quad (20)$$

Similar considerations hold for the maximum:

$$\overline{P}(c|\mathbf{a}) = \sum_{c' \in \mathcal{C} \setminus \{c\}} \frac{n(c') + s \tilde{\delta}_{c'\tilde{c}}}{n(c) + \frac{\epsilon}{|\mathcal{C}|}} \left(\frac{n(c) + s}{n(c') + s} \right)^k \cdot \prod_{i=1}^k \frac{n(a_i, c') + s[\frac{\epsilon}{|\mathcal{A}_i|} + 1 - \epsilon]}{n(a_i, c) + s \frac{\epsilon}{|\mathcal{A}_i|}}, \quad (21)$$

with

$$\tilde{\delta}_{c'\tilde{c}} := \begin{cases} \frac{\epsilon}{|\mathcal{C}|} + (1 - \epsilon) & \text{if } c' = \tilde{c} \\ \frac{\epsilon}{|\mathcal{C}|} & \text{otherwise} \end{cases}.$$

Appendix B: Lower marginal

Consider the evaluation of the lower probability for the joint instance of the attributes defined as in (13). This corresponds to evaluate the minimum of the expression in (12), again with the probabilities defined as in (16) and (17) with respect to the constraints in (18) and (19).

When minimizing this objective function, we first consider the (trivial) minimization with respect to the constraints in (19). Then, we obtain the following linear program:

$$\text{minimize:} \quad \sum_{c \in \mathcal{C}} \left[\prod_{i=1}^n \frac{n(a_i, c) + s \frac{\epsilon}{|\mathcal{A}_i|}}{n(c) + s} \right] t_c \quad (22)$$

$$\text{w.r.t.:} \quad \frac{\epsilon}{|\mathcal{C}|} \leq t_c \leq \frac{\epsilon}{|\mathcal{C}|} + (1 - \epsilon) \quad (23) \\ \sum_{c \in \mathcal{C}} t_c = 1,$$

whose solution $\{t_c^*\}$ can be plugged in (12) to obtain (13).

ProbModelXML. A format for encoding probabilistic graphical models

Manuel Arias

Dept. Artificial Intelligence. UNED. Madrid, Spain

Francisco Javier Díez

Dept. Artificial Intelligence. UNED. Madrid, Spain

Miguel Palacios-Alonso

Computer Science Dept. INAOE, Tonantzintla, Mexico

Íñigo Bermejo

Dept. Artificial Intelligence. UNED. Madrid, Spain

Abstract

ProbModelXML is an XML format for encoding probabilistic graphical models. The main advantages of this format are that it can represent several kinds of models, such as Bayesian networks, Markov networks, influence diagrams, LIMIDs, decision analysis networks, as well as temporal models: dynamic Bayesian networks, MDPs, POMDPs, Markov processes with atemporal decisions (MPADs), DLIMIDs, etc., and the possibility of encoding new types of networks and user-specific properties without the need to modify the format definition.

1 Introduction

A probabilistic graphical model (PGM) consists of a probability distribution and a graph, such that each node in the graph represents one of the variables on which the probability is defined, and the structure of the graph imposes some properties of independence on the probability distribution. Some PGMs are purely probabilistic, such as Bayesian networks (Pearl, 1988), while others, such as influence diagrams (Howard and Matheson, 1984), include decisions and utilities. Dynamic PGMs (Dean and Kanazawa, 1989; Murphy, 2002) are temporal models that discretize time in intervals of a fixed duration (cycle length) and create an instance of each variable for each time period.

Several formats have been developed for encoding PGMs, but almost all of them are designed for a single software tool. One exception is Fabio Cozman's XMLBIF (see Sec. 4.1), that has been implemented by several software tools, but it is restricted to Bayesian networks containing only discrete variables, with a very limited set of features. Another exception was DSC, proposed by Microsoft as a standard format for Bayesian networks and influence diagrams, that would receive contributions from the UAI (uncertainty in artificial intelligence) community; however, some time later Microsoft

removed the web pages of DSC and developed a new XML format, MSBNx, limited to Bayesian networks.

For this reason, we decided to develop a new format for encoding PGMs, that presents two main advantages with respect to previous proposals. First, it can encode several types of PGMs: its current version includes Bayesian networks, Markov networks, influence diagrams, LIMIDs, decision analysis networks, dynamic Bayesian networks, MDPs, Markov processes with atemporal decisions (MPADs), POMDPs, Dec-POMDPs, and DLIMIDs (see (Arias et al., 2011) for definitions and references), and it also permits to encode new models by combining the existing *constraints* or by defining new ones (see Sec. 3.1.1). The second advantage is that it can encode user-specific features by using the *AdditionalProperties* tag (see Sec. 3.1.2). ProbModelXML was designed as the default format for OpenMarkov,¹ an open-source tool for probabilistic graphical models, but our purpose is not only to fulfill the needs of a single tool and a single research group, but to offer an extensible well-documented format that can be used by a large community of people. This is the main reason for submitting this paper to the PGM workshop: to

¹www.openmarkov.org.

make the format known to a wide audience and to receive suggestions from our colleagues. In fact, we will not release version 1.0.0 of the format until we have received the feedback from the communities of PGMs and POMDPs.

The rest of the paper is structured as follows: in Section 2.1 defines the basic properties of PGMs, including dynamic models. Section 3 describes the most relevant aspects of the specification of the format. Section 4 reviews other formats for PGMs and Section 5 contains the conclusions. Given that it is impossible to present in this paper all the details of the format, we will limit ourselves to describing its main features. The complete specification can be found in (Arias et al., 2011).

2 Background: Probabilistic graphical models

2.1 Basic properties

A probabilistic graphical model (PGM) consists of a set of variables \mathbf{V} , a graph \mathcal{G} such that each node represents a variable in \mathbf{V} , and a probability distribution P that satisfies certain properties of independence dictated by the structure (the links) of the graph (Pearl, 1988). In some PGMs all the nodes represent chance variables; in this case, the probability distribution is defined over \mathbf{V} : $P(\mathbf{v})$. Other models contain three types of nodes: chance, decision, and utility, denoted by \mathbf{C} , \mathbf{D} , and \mathbf{U} , respectively, such that $\mathbf{V} = \mathbf{C} \cup \mathbf{D} \cup \mathbf{U}$; in this case, the probability distribution is $P(\mathbf{c}|\mathbf{d})$.

There are three types of variables, depending on their domain. A variable is *finite-states* if it takes values on finite set of values. It is *numeric* if it represents the result of a measurement, including the count of the number of elements in a set. A *discretized* variable is a numeric variable that has been assigned a finite set of thresholds, which induce a finite set of intervals; each interval is considered as a state of the variable.

Links can be directed or undirected. Paths can be open or closed. If a closed path can be traversed completely crossing all its directed links forwards (i.e., from the tail of the link to its head), then that path is a *cycle*; otherwise it is a *loop*—see the examples in (Arias et al., 2011). A *self-loop* is a link whose nodes are the same; for example, $A \rightarrow A$ or $A \rightarrow A$. It can also be defined as a closed path consisting of only one link. Most PGMs do not accept self-loops, but there are exceptions, such as the representation of dynamic models in Netica and GeNIE.²

²See www.norsys.com and genie.sis.pitt.edu.

2.2 Dynamic models

Dynamic PGMs are a generalization of some Markovian models proposed several decades earlier. Thus, dynamic Bayesian networks (Dean and Kanazawa, 1989; Murphy, 2002) extend Markov chains and hidden Markov models, by allowing that the state of the system be represented by a set of variables rather than by a single variable. In the same way, Markov Decision Process (MDPs) (Bellman, 1957) are extended by factored MDPs (Boutilier et al., 1995; Boutilier et al., 2000)) and Partially Observable Markov Decision Process (POMDPs) are extended by factored POMDPs (Boutilier and Poole, 1996)). DLIMIDs (Díez and van Gerven, 2011) are very similar to POMDPs, but they allow several decision nodes per time slice.

3 The ProbModelXML format

Each ProbModelXML file may contain one network and, optionally, some *inference options*, such as the default inference algorithm or an elimination ordering. Alternatively, a file may contain some *evidence*, consisting of one or several evidence cases, each one composed of a set of *findings*. A third possibility is that the file contains a set of *policies*; it may be a strategy returned by an algorithm or a policy imposed by the user to perform what-if reasoning.

Therefore, the skeleton of a file in ProbModelXML format is:

```
<?xml version="1.0" encoding="UTF-8"?>
<ProbModelXML formatVersion="1.0">
  <ProbNet />0..1
  <Policies />0..1
  <InferenceOptions />0..1
  <Evidence />0..1
</ProbModelXML>
```

The default encoding for ProbModelXML is UTF-8, the same as for XML. Therefore, it is redundant to write encoding="UTF-8", but we prefer to say it explicitly, because it is possible to use other encodings.

3.1 Specification of probabilistic networks

The skeleton for a probabilistic network is as follows:

```
<ProbNet type=enumNetworkType >
  <AdditionalConstraints />0..1
  <Comment />0..1
  <Language />0..1
  <AdditionalProperties />0..1
  <Variables />
```

```
<Links />
<Potentials />
</ProbNet>
```

In the current version of the format, the type can be one of the following: *BayesianNetwork* (Pearl, 1988), *MarkovNetwork* (Pearl, 1988), *InfluenceDiagram* (Howard and Matheson, 1984), *LIMID* (Lauritzen and Nilsson, 2001), *DynamicBayesianNetwork* (Dean and Kanazawa, 1989; Murphy, 2002), *MarkovDecisionProcess* (which includes factored MDPs (Boutilier et al., 2000)), *POMDP* (which includes factored POMDPs (Boutilier and Poole, 1996) and MOMDPs (Ong et al., 2009)), and *DLIMID* (Díez and van Gerven, 2011).

3.1.1 Constraints

The main purpose of constraints is to prevent the user from doing illegal operations at the graphical user interface (GUI), such as giving an empty name to a variable or creating a cycle in a Bayesian network. Another use of constraints may be, for example, to prevent a learning algorithm from adding more than n parents to a node.

In the OpenMarkov tool, each network type is defined by a set of basic constraints. For example, one of the constraints that define a Bayesian network is that it can not contain cycles; another constraint is that it can not contain decision nor utility nodes. The basic constraints available in ProbModelXML and the set of constraints that define each network type are listed in (Arias et al., 2011). Decoupling the constraints from the rest of the code has facilitated the treatment of new network types in the future.

Additionally, the user can assign to a network other constraints that are not imposed by the network type; for example, the constraint that the network contains only finite-state variables. This way, an algorithm that can only solve Bayesian networks with finite-state variables may check that the network has that constraint assigned (an immediate check) or that all the variables are finite-states (which would require examining all the variables). If the network does not satisfy this constraint, the algorithm will throw an exception, explaining why it can not evaluate that network. Obviously, other software tools can treat constraints in different ways.

3.1.2 Additional properties

This tag permits to extend the ProbModelXML format by representing user-defined properties. This tag can appear in the context of *ProbNet* (see

Section 3.1), *Variable*, *State*, *Potential*, *EvidenceCase*, and *Policy* (see below). In all cases, its skeleton is:

```
<AdditionalProperties>
  <Property name=string value=string/>1..n
</AdditionalProperties>
```

The main use of this tag is to store properties that are not part of the ProbModelXML format. For example, in Elvira each variable has a name, which is a string with some restrictions, and a title. In ProbModelXML we do not need this distinction, because the name can be any string. If the software package Elvira had to encode a variable in ProbModelXML, the name can be stored as the attribute name, while the title can be stored in the list of *AdditionalProperties*: `<Property name="elvira.title" value="Test result"/>`. Similarly, in GeNIE each node has a fill-in color, that can be stored as follows: `<Property name="genie.interior.color" value="e5f6f7"/>`. This way, any tool can save its models in ProbModelXML without missing information. Furthermore, a second tool might read that network, storing apart the properties that it does not understand, and edit it; when saving the network, it can write down also the properties stored apart. Then, the first tool can open again the modified network, thus recovering the properties that were not understood by the second.

Another use of the *AdditionalProperties* tag is to encode a multi-lingual version of the network. For example, if the network has a property `<Language>en</Language>` (cf. Sec. 3.1), which means that its default language is English, we may have the following variable:

```
<Variable name="Fever">
  ...
  <AdditionalProperties>
    <Property name="name.es" value="Fiebre"/>
    <Property name="name.fr" value="Fièvre"/>
    <Property name="name.de" value="Fieber"/>
  </AdditionalProperties>
<Variable>
```

When the network is displayed in English, the name of this variable will appear as "Fever", when in Spanish, as "Fiebre", etc.

3.1.3 Variables

The skeleton for encoding a variable is:

```
<Variable name=string type=enumDomainType
  role=enumNodeRole0..1
  <Coordinates />0..1
```

```
<AdditionalProperties />0..1
specification_of_domain
</Variable>
```

The type is an enumerate that can take on three values: *FiniteStates*, *Numeric*, and *Discretized*. The role can be *Chance*, *Decision*, or *Utility*. The domain of a finite-states variable is specified by a list of *States*:

```
<States>
  <State name=string>
    <AdditionalProperties />0..1
  </State>2..n
</States>
```

The domain of a numeric variable is specified by two thresholds, that define an interval, plus a number that denotes the precision with which its value is measured.

```
<Unit>string</Unit>0..1
<Precision>decimalNumber</Precision>
<Interval>
  <Threshold value=number
    belongsTo=enumSide/>2
</Interval>
```

Given that a discretized variable can be viewed as being finite-states and numeric at the same time, its skeleton has the tags of both:

```
<Unit />0..1
<Precision />
<Thresholds>
  <Threshold />3..n
</Thresholds>
<States />2..n
```

3.1.4 Links

The skeleton of a link is:

```
<Links>
  <Link var1=string var2=string directed=true>
    <Comment />0..1
    <Label>string</Label>
    <AdditionalProperties />0..1
  </Link>0..n
</Links>
```

The *AdditionalProperties* tag can be used to declare new types of links. For example, when learning a Bayesian network from a database we can use a *model network* that determines which links must be necessarily present in the learned network, which links are forbidden, etc. This can be accomplished by assigning *AdditionalProperties* to the links in the model network.

3.1.5 Potentials

A potential ψ defined on a set of variables \mathbf{V} is a function that assigns a real number to each configuration \mathbf{v} of \mathbf{V} . The skeleton of a potential in ProbModelXML is:

```
<Potential type=enumPotentialType
  role=enumPotentialRole label=string>
  <Comment />0..1
  <AdditionalProperties />0..1
  specification_of_the_potential
</Potential>
```

The role of a potential can be *JointProbability*, *ConditionalProbability*, *Utility*, or *Policy*.

In the current version of the format we have several types of potentials. A *Table* can be used when all the variables are finite-states or discretized; essentially, it consists of a list of numeric values (parameters), one for each configuration of the variables; it is also possible to assign second-order probability distributions to the parameters: *Beta*, *Dirichlet*, *Normal*, *LogNormal*, etc., that can be used for sensitivity analysis.

A *Tree/ADD* is a compact way of storing tables in which some of the numeric values repeat themselves; each branch outgoing from a finite-states variable corresponds to one or several of its states; each branch outgoing from a numeric variable corresponds to an interval. The difference between a tree and an ADD (algebraic decision diagram (Babar et al., 1993)) is that the structure of the latter is an acyclic directed graph. In ProbModelXML there is only one potential type for both trees and ADDs.

ICIModel (where ICI stands for “independence of causal influence”) is the type of potential used to represent canonical models (Díez and Drudzel, 2006), such as the noisy-OR, noisy-MAX, etc. It contains a potential for each link in the family and, optionally, another subpotential for the leak probability.

```
<Potential type="ICIModel"
  role="ConditionalProbability" >
  <Model>or</Model>
  <Subpotentials>
    ...
  </Subpotentials>
</Potential>
```

The *LinearCombination* potential is used when a **numeric** variable Y depends **deterministically** on a mixed set of variables $\mathbf{X} = \mathbf{D} \cup \mathbf{C}$ —which in general will be the parents of node Y in the graph—where \mathbf{C} is the set of numeric variables and \mathbf{D} is

the set of finite-states variables:

$$y = \alpha(\mathbf{d}) + \sum_{i=1}^m \beta_i(\mathbf{d}) \cdot c_i .$$

This function allows the parents of a utility node to be any combination of chance, decision, and utility nodes, a feature that we missed when building an influence diagram for a medical problem, because standard influence diagrams do not admit this possibility.

In a *ConditionalGaussian* potential (Lauritzen and Wermuth, 1989), the conditional probability density for Y is a univariate normal distribution:

$$f(y|\mathbf{d}, \mathbf{c}) \sim \mathcal{N}(\mu(\mathbf{d}, \mathbf{c}), \sigma^2(\mathbf{d})) ,$$

where

$$\mu(\mathbf{d}, \mathbf{c}) = \alpha(\mathbf{d}) + \sum_{i=1}^m \beta_i(\mathbf{d}) \cdot c_i .$$

The potentials *Exponential* and *MixtureOfExponentials*, used in combination with *Tree/ADD*, can represent mixtures of truncated exponentials (Moral et al., 2001). Other potentials available in ProbModelXML are *LogisticRegression* and *Delta*.

3.2 Specification of evidence

A **finding** is the assignment of a value to a variable. A set of findings is an **evidence case**. Therefore, the skeleton for evidence is:

```
<Evidence>
  <EvidenceCase>
    <Finding variable="string" state="string"
      stateIndex="integer"
      numericValue="number"/>0..n
  </EvidenceCase>0..n
</Evidence>
```

The attribute variable is compulsory. If the variable is of type finite-states, then either the state or stateIndex must be present. If the variable is numeric, then numericValue must be present. If the variable is discretized, only one of the attributes may be present.

3.3 Specification of policies

In an influence diagram, a policy for a decision D is the assignment of a probability distribution to each configuration of the informational predecessors of D , i.e., the variables whose values are known when making the decision. A deterministic policy is the assignment of a degenerate distribution to each configuration. Therefore, the specification of a policy is essentially the same as that of a conditional probability potential (cf. Sec. 3.1.5).

3.4 Specification of dynamic models

The representation of dynamic models in their compact form (see Sec. 2.2) is similar to that of other types of models. There are, however, specific network tags, such as *TimeUnit*, *CycleLength*, *Horizon*, and *CoordinatesShift*, which is used to determine the relative positions of temporal variables in different time slices.

Temporal variables are recognized because they have an attribute indicating the time slice that contains the variable:

There are also specific potentials for dynamic models, such as *CycleLengthShift* and *WeibullDistribution*.

4 Related work: other formats

Several formats have been developed for probabilistic graphical models (PGMs) and Markov decision processes (MDPs). In this section we review briefly those that are more related to ProbModelXML.

4.1 Formats for Bayesian networks and influence diagrams

DNET (Netica) DNET was developed by Norsys Software Corp. as the default format for their software package, Netica.³ This format can represent Bayesian networks, influence diagrams, MDPs and POMDPs, with both discrete (finite-states) or continuous variables. Its specification is available at www.norsys.com/downloads/DNET_File_Format.txt.

Elvira Elvira (Elvira Consortium, 2002) started in 1997 as a joint project of several Spanish universities.⁴ The Elvira format, which uses a C-like syntax, can represent Bayesian networks and influence diagrams with continuous and finite-state variables, canonical models (Díez and Druzdzel, 2006), uncertain parameters (defined by intervals), etc. Its specification can be found at leo.ugr.es/elvira/devel/Formato/formato.html. Some of the features of the Elvira format were inspired on DNET, and in turn many of ProbModelXML's features are borrowed from Elvira.

XMLBIF It was proposed by Fabio Cozman, with suggestions from Marek Druzdzel and others—see www.poli.usp.br/p/fabio.cozman/Research/InterchangeFormat. It is restricted to the representation of Bayesian networks with finite-state

³See www.norsys.com/netica.html.

⁴leo.ugr.es/elvira and www.ia.uned.es/~elvira.

variables. It is the default format for Cozman’s JavaBayes tool.⁵ Weka⁶ and many of the tools for PGMs can read and write networks in this format.

MSBNx This XML format was proposed by Microsoft as the default format for their Microsoft Bayesian Network (MSBNx) tool, as a replacement for the old non-XML format DSC. It can only represent Bayesian networks. See research.microsoft.com/en-us/um/redmond/groups/adapt/msbnx and research.microsoft.com/en-us/um/redmond/groups/adapt/msbnx/msbnx/File_Formats.htm.

XDSL It is the default format for SMILE and GeNIE, two programs developed by the Decision Systems Laboratory at the University of Pittsburgh.⁷ It can represent Bayesian networks, influence diagrams, and some dynamic models, with both continuous and finite-state variables. It can also represent canonical models. The XML schemas (XSDs) that define it are available at genie.sis.pitt.edu/SMILEHelp/Appendices/XDSL_File_Format_-_XML_Schema_Definitions.htm.

4.2 Formats for POMDPs

Cassandra’s format Anthony Cassandra has used a format for flat (i.e., non-factored) POMDPs, that is available at www.cassandra.org/pomdp/code/pomdp-file-grammar.shtml and www.cassandra.org/pomdp/code/pomdp-file-spec.shtml—see also www.cassandra.org/pomdp/examples. The software package Perseus (Spaan and Vlassis, 2005) can also read files in Cassandra’s format.

SPUDD It is the format used by the software package SPUDD, which stands for “Stochastic Planning using Decision Diagrams” (Hoey et al., 1999). It can encode factored MDPs and POMDPs. Potentials are represented by algebraic decision diagrams (ADDs)—see Section 3.1.5. Some example POMDPs (files having the extension .txt) are included in a tar.gz file, together with SPUDD’s C++ source code, which is available at www.computing.dundee.ac.uk/staff/jessehoey/spudd/index.html.

The tool Symbolic Perseus,⁸ by Pascal Poupart (Poupart, 2005), uses a subset of the SPUDD format, whose grammar is specified in this file:

⁵www.pmr.poli.usp.br/ltd/Software/javabayes.

⁶www.cs.waikato.ac.nz/ml/weka.

⁷See genie.sis.pitt.edu.

⁸See www.cs.uwaterloo.ca/~ppoupart/software.html. The file ParseSPUDD.java contains the parser for the SPUDD format.

www.cs.uwaterloo.ca/~ppoupart/software/symbolicPerseus/problems/SYNTAX.txt—see also the examples at www.cs.uwaterloo.ca/~ppoupart/software/symbolicPerseus/problems.

PomdpX It is an XML format for POMDPs, developed at the University of Singapore: bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n>Main.PomdpXDocumentation. It admits flat POMDPs, as well as MOMDPs. In a MOMDP the state space is factored into two variables, X (observable) and Y (unobservable), and there is a third variable in each time slice, O , which provides indirect information about Y (Ong et al., 2009). There is a companion XML format for representing the policy obtained when evaluating a POMDP: bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n>Main.PolicyXDocumentation.

PPDDL and RDDL Two additional formats were proposed to encode the problems proposed at the Probabilistic Planning Track of the International Planning Competition (IPC). The Probabilistic Planning Domain Definition Language (PPDDL),⁹ used at the 4th and 5th IPC in 2004 and 2006 respectively, was able to encode factored MDPs with finite-state variables. The Relational Dynamic Influence Diagram Language (RDDL), used at the 7th IPC in 2011, was able to represent relational (PO)MDPs with both finite-state and continuous variables.¹⁰ These formats were not intended to encode non-temporal Bayesian networks and influence diagrams, but they have enough expressive power to represent them as well.

4.3 Discussion

We have seen that most formats proposed so far have important limitations: each format can represent either Bayesian networks (sometimes in conjunction with influence diagrams) or POMDPs, with the exception of Netica’s DNET, which claims to admit both types of models, but it uses a non-standard representation of POMDPs, which makes it incompatible with existing packages for POMDPs.

A drawback of many of these formats is that they use a syntax inspired on C++ or Lisp, which complicates the creation of parsers for them. In contrast, there are efficient tools for writing parsers

⁹www.tempastic.org/papers/CMU-CS-04-167.pdf.

¹⁰users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf.

that can read XML files into C++ or Java programs. An inconvenience of XML is its verbosity, which leads to significantly larger files than when using a C-like syntax. This would be relevant if a user had to encode the information—a PGM, in our case—into an XML format manually. Fortunately, there are many computer programs that facilitate that task; in our case, it is possible to use OpenMarkov’s GUI to build PGMs without caring about the syntax of the format, and we expect that other software tools may adopt this format in the future. Given that the simplicity of parsing XML files compensates for the size of the files, XML is used more and more for defining new formats in almost every field of computing. However, two of the three XML formats proposed previously for PGMs can only encode Bayesian networks with finite-state variables, the third can only encode Bayesian networks and influence diagrams, and the only XML format for POMDPs cannot encode factored models.

Finally, an important limitation of all these formats is that they are not extensible, which implies that they cannot encode any property that has not been explicitly declared in its specification.

5 Conclusion

In this paper we have presented an overview of a new XML format for encoding probabilistic graphical models (PGMs). One of its advantages is the possibility of representing several types of models, such as Bayesian networks, Markov networks, influence diagrams, LIMIDs, as well as dynamic models: dynamic Bayesian networks, MDPs, POMDP, and DLIMIDs, and it is easy to add new types of models by combining the existing constraints or by defining new ones.

Another advantage with respect to existing formats is the possibility of encoding user-defined properties without modifying the specification of the format, by placing them under the `AdditionalProperties` tag. The third advantage is its XML syntax, which permits to use the utilities available for generating parsers and writers in different languages: Java, C++, etc.

For these reasons, we believe that ProbModelXML may be very useful as an interchange format for PGMs. Clearly, ProbModelXML is a very rich format, which makes it difficult to implement all its features. However, each software package can implement only the subset of features required for its purpose: it suffices to throw an error message when the parser encounters an unknown feature. Even our software tool OpenMarkov is currently

unable to cope with several features of the format, but we have decided to include them in the format to satisfy the needs of other research groups. For this reason, we believe that the format presented in this paper can be very useful for interchanging several types of PGMs between different software tools and research groups.

The tasks we have scheduled for the near future are to improve the syntax for certain properties in the light of the feedback we have received from some colleagues, and to extend it to cover new types of potentials (such as mixtures of polynomials (Shenoy, 2011; Shenoy and West, 2010)), submodels (as in GENIE), and new types of networks, such as object-oriented Bayesian networks (Koller and Pfeffer, 1997) and probabilistic relational models (Jaeger, 1997; Koller and Pfeffer, 1996).

Acknowledgments

This work has been supported by grants TIN2006-11152 and TIN2009-09158, of the Spanish Ministry of Science and Technology, and by FONCICYT grant 85195. I.B. has received a predoctoral fellowship from the Universidad Nacional de Educación a Distancia (UNED).

The reviewers of the PGM-2012 workshop have made useful comments about this paper.

References

- [Arias et al.2011] M. Arias, F. J. Díez, and M. P. Palacios. 2011. ProbModelXML. A format for encoding probabilistic graphical models. Technical Report CISIAD-11-02, UNED, Madrid, Spain.
- [Bahar et al.1993] R. I. Bahar, E. A. Frohm, C. M. Gaona, et al. 1993. Algebraic decision diagrams and their applications. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD’93)*, pages 188–191, Santa Clara, CA.
- [Bellman1957] R. E. Bellman. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.
- [Boutilier and Poole1996] C. Boutilier and D. Poole. 1996. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1168–1175, Portland, OR. AAAI Press.
- [Boutilier et al.1995] C. Boutilier, R. Dearden, and M. Goldszmidt. 1995. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1104–1111, Montreal, Canada.

- [Boutilier et al.2000] C. Boutilier, R. Dearden, and M. Goldszmidt. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121:49–107.
- [Dean and Kanazawa1989] T. Dean and K. Kanazawa. 1989. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150.
- [Díez and Druzdzel2006] F. J. Díez and M. J. Druzdzel. 2006. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01, UNED, Madrid, Spain.
- [Díez and van Gerven2011] F. J. Díez and M. A. J. van Gerven. 2011. Dynamic LIMIDs. In L. E. Sucar, J. Hoey, and E. Morales, editors, *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, pages 164–189. IGI Global, Hershey, PA.
- [Elvira Consortium2002] The Elvira Consortium. 2002. Elvira: An environment for creating and using probabilistic graphical models. In J. A. Gámez and A. Salmerón, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM'02)*, pages 1–11, Cuenca, Spain.
- [Hoey et al.1999] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 279–288, Stockholm, Sweden. Morgan Kaufmann, San Francisco, CA.
- [Howard and Matheson1984] R. A. Howard and J. E. Matheson. 1984. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, pages 719–762. Strategic Decisions Group, Menlo Park, CA.
- [Jaeger1997] M. Jaeger. 1997. Relational Bayesian networks. In *Proceedings of the Thirteenth Conference in Artificial Intelligence (UAI-97)*, pages 266–273, San Francisco, CA. Morgan Kaufmann.
- [Koller and Pfeffer1996] D. Koller and A. Pfeffer. 1996. Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 580–587, Madison, WI.
- [Koller and Pfeffer1997] D. Koller and A. Pfeffer. 1997. Object-oriented Bayesian networks. In *Proceedings of the Thirteenth Conference in Artificial Intelligence (UAI-97)*, pages 302–313, San Francisco, CA. Morgan Kaufmann.
- [Lauritzen and Nilsson2001] S. L. Lauritzen and D. Nilsson. 2001. Representing and solving decision problems with limited information. *Management Science*, 47:1235–1251.
- [Lauritzen and Wermuth1989] S. L. Lauritzen and N. Wermuth. 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17:31–57.
- [Moral et al.2001] S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. *Lecture Notes in Artificial Intelligence*, 2143:156–167.
- [Murphy2002] K. Murphy. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, Computer Science Division, University of California, Berkeley.
- [Ong et al.2009] S. C.W. Ong, S. W. Png, D. Hsu, and W. S. Lee. 2009. POMDPs for robotic tasks with mixed observability. In *Proceedings of Robotics: Science and Systems V*, Seattle, WA.
- [Pearl1988] J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.
- [Poupart2005] P. Poupart. 2005. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. Ph.D. thesis, Dept. of Computer Science, University of Toronto, Canada.
- [Shenoy and West2010] P. P. Shenoy and J. C. West. 2010. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52:641–657.
- [Shenoy2011] P. P. Shenoy. 2011. A re-definition of mixtures of polynomials for inference in hybrid Bayesian networks. In W. Liu, editor, *Proceedings of the 11th European conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'11)*, pages 98–109. Springer, Heidelberg.
- [Spaan and Vlassis2005] M. T. J. Spaan and N. Vlassis. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220.

Gaussian Join Tree classifiers with applications to mass spectra classification

Victor Bellón,

Universitat de Barcelona, Spain

IIIA-CSIC, Spain

bellon@ub.edu

Jesús Cerquides

IIIA-CSIC, Spain

cerquide@iiia.csic.es

Ivo Grosse

Martin-Luther-Universität, Germany

grosse@informatik.uni-halle.de

Abstract

Classifiers based on probabilistic graphical models are very effective. In continuous domains, parameters for those classifiers are usually adjusted by maximum likelihood. When data is scarce, this can easily lead to overfitting. Nowadays, models are sought in domains where the number of data items is small and the number of variables is large. This is particularly true in the realm of bioinformatics. In this work we introduce Gaussian Join Trees (GJT) classifiers to try to partially overcome this issue by performing exact bayesian model averaging over the parameters. We use two different mass spectra classification datasets for cancer prediction to compare GJT classifiers with those learnt by maximum likelihood.

1 Introduction

Supervised classification is a basic task in data analysis and pattern recognition. It requires the construction of a classifier, that is, a function that assigns a class label to instances described by a set of variables. There are numerous classifier paradigms, among which the ones based on probabilistic graphical models (PGMs) (Lauritzen, 1996), are very effective and well-known in domains with uncertainty.

A widely used assumption is that data follows a multidimensional Gaussian distribution(Geiger and Heckerman, 1994). This is adapted for classification problems by assuming that data follows a multidimensional Gaussian distribution that is different for each class, encoding the resulting distribution as a Conditional Gaussian Network (CGN)(Böttcher, 2004). In (Larrañaga et al., 2006), Larrañaga,

Pérez and Inza introduce and evaluate classifiers based on CGNs with a more detailed description in (Pérez, 2010). They analyze different methods to identify a Bayesian network structure and a set of parameters such that the resultant CGN performs well in the classification task. In (Pérez, 2010) the authors propose to estimate the parameters directly from the sample mean and sample covariance matrix in the data, that is using maximum likelihood. Following this strategy can lead to model overfitting when data is scarce. In bioinformatics, models are sought in domains where the number of data items is small and the number of variables is large, such as classification of mass spectrograms or microarrays. To try to avoid overfitting, we introduce classifiers based on Gaussian Join Trees (GJT) that instead of estimating by maximum likelihood perform exact Bayesian av-

eraging over the parameters.

We start by reviewing hyper Markov laws (Dawid and Lauritzen, 1993), and the hyper inverse Wishart distribution in section 2. Those results are needed to introduce in section 3.1 the hyper normal inverse Wishart (\mathcal{HNIW}) distribution, and prove that it is a strong hyper Markov law. This means that Bayesian learning and inference can be performed locally and efficiently. The application of the \mathcal{HNIW} to build GJT classifiers is provided in section 3.2 and a preliminary empirical comparison is presented in section 4.

The contributions of the work are: (i) the proposal of GJT classifiers including a proof that the \mathcal{HNIW} law is strong hyper Markov, (ii) a preliminary empirical comparison with classifiers adjusting parameters by maximum likelihood for the task of mass spectra classification and, (iii) an open source implementation of the algorithms, made available for easy reproducibility of the results. Furthermore, in order to prove that the \mathcal{HNIW} is strong hyper Markov we need to determine the marginals of a normal inverse Wishart distribution, a result that we have not found elsewhere in the literature.

2 Overview of hyper Markov laws

In this section we succinctly review the fundamental results in (Dawid and Lauritzen, 1993). The interested reader can find some of the missing definitions and additional details in (Dawid and Lauritzen, 1993).

2.1 Markov distributions over decomposable graphs

In the following, let $\mathcal{G} = (V, E)$ be an undirected decomposable graph over a set of random variables. A graph is said to be decomposable when all of its prime components are complete subgraphs of \mathcal{G} ; we refer to all maximal prime components as cliques of the graph.

Definition 1. A distribution P on V is called Markov over \mathcal{G} if for any decomposition (A, B) of \mathcal{G}

$$A \perp\!\!\!\perp B | A \cap B.$$

A graphical model $M(\mathcal{G})$ is a family of probability distributions which are Markov over \mathcal{G} .

Definition 2. We say that distributions Q over A and R over B are consistent if both yield the same distribution over $A \cap B$.

The next theorem tells us that given a set of marginal probability distributions over the cliques of a graph that are consistent between them, we can assess the unique joint probability distribution having those marginals. Let \mathcal{C} be the set of cliques of \mathcal{G} , and \mathcal{S} the corresponding collection of separators (including possible repetitions) (see (Dawid and Lauritzen, 1993; Cowell et al., 1999) for details).

Theorem 1. *Given a pairwise consistent collection of distributions $\{Q_C : C \in \mathcal{C}\}$, Q_C being a distribution over C , the unique Markov distribution over \mathcal{G} having $\{Q_C\}$ as its marginals is*

$$p(x) = \frac{\prod_{C \in \mathcal{C}} p_C(x_C)}{\prod_{S \in \mathcal{S}} p_S(x_S)}, \quad (1)$$

where p_C , the marginal of p over the clique C is Q_C and p_S is the marginal of any of the cliques in which the separator S is included.

2.2 Hypermarkov laws

Let θ be a quantity parameterising a graphical model $M(\mathcal{G})$. A hyper Markov law is then defined by a property which mimics the global Markov property, at the parameter level

Definition 3. A law on $M(\mathcal{G})$ is called (weak) hyper Markov over \mathcal{G} if for any decomposition (A, B) of \mathcal{G}

$$\theta_A \perp\!\!\!\perp \theta_B | \theta_{A \cap B}$$

Definition 4. A law on $M(\mathcal{G})$ is called strong hyper Markov over \mathcal{G} if for any decomposition (A, B) of \mathcal{G}

$$\theta_{B|A} \perp\!\!\!\perp \theta_A$$

The family of hyper Markov laws and the family of strong hyper Markov laws each form a conjugate family for the sampling family $M(\mathcal{G})$. Strong hyper Markov laws produce an especially simple decomposition of the Bayesian analysis into a collection of subanalyses for smaller problems. Thus, the posterior after observing some data can be assessed locally.

The next two propositions from (Dawid and Lauritzen, 1993) will be needed later to prove that the hyper normal inverse Wishart distribution is strong hyper Markov.

Proposition 1. *Given a set of hyperconsistent laws $\{\mathcal{M}_C\}$ over clique marginals, there is a unique hyper Markov law over \mathcal{G} satisfying those marginals, which is called the hyper Markov combination of $\{\mathcal{M}_C\}$.*

Proposition 2. *Let $\mathcal{P} \subseteq M(\mathcal{G})$ be a subfamily of the Markov models over \mathcal{G} . Assume that \mathcal{P} is weak meta Markov, and for any complete set S in \mathcal{G} the model \mathcal{P}_S form a full exponential family. Let \mathcal{L} be a hyper Markov law such that, for any clique C , the law of θ_C is a conjugate prior distribution for the model \mathcal{P}_C . Then \mathcal{L} is strong hyper Markov.*

2.3 Hyperinverse Wishart distribution

Let \mathcal{G} be a decomposable graph over a set of continuous variables. We are interested in the subfamily of models which are in $M(\mathcal{G})$ and which are assumed to be jointly multivariate normal with mean equal to zero and unknown positive definite covariance matrix Σ , that is $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$. For this particular family, it is possible to define a strong hyper Markov distribution. It was introduced in (Dawid and Lauritzen, 1993), under the name of hyper inverse Wishart distribution

Definition 5. Let $\Phi^C = \{\Phi^C, C \in \mathcal{C}\}$ be a collection of positive definite dispersion matrices, where Φ^C is the dispersion matrix over the variables in clique C . Assume that the matrices are consistent, that is, for each $B \subseteq C_1 \cap C_2$, the sub-matrices¹ $\Phi^{C_1}[B, B]$ and $\Phi^{C_2}[B, B]$ are identical. Let δ be a positive real number. We can define a collection of hyper consistent Markov laws by defining over each matrix Σ^C the following law:

$$\mathcal{L}(\Sigma^C) = \mathcal{IW}(\delta; \Phi^C).$$

The law that results from the hyper Markov

¹Given a matrix M and sets of indexes I, J , we note $M[I, J]$ the sub-matrix that keeps the rows with indexes in I and the columns with indexes in J . Equivalent notation is used for vectors.

combination of this collection of laws is called hyper inverse Wishart and noted $\mathcal{HIW}(\delta, \Phi^C)$.

The \mathcal{HIW} is proved strong hyper Markov in (Dawid and Lauritzen, 1993).

3 Gaussian join tree classifiers

In this section we introduce Gaussian join tree classifiers as an alternative to conditional Gaussian network classifiers in (Larrañaga et al., 2006; Pérez, 2010). We start by introducing a new hyper Markov distribution, the hyper normal inverse Wishart distribution, that generalizes the Hyper inverse Wishart distribution in the case when the mean is also unknown. After that, we provide the algorithm to perform learning and inference using these distributions.

3.1 Hyper normal inverse Wishart distribution

\mathcal{HIW} laws can only be used when the multivariate mean is known to be $\mathbf{0}$. We are interested in the more general subfamily of models in $M(\mathcal{G})$ with any possible mean, that is $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. We introduce the hyper normal inverse Wishart distribution and prove that it is strong hyper Markov.

Definition 6. Let $\Phi^C = \{\Phi^C, C \in \mathcal{C}\}$ be a collection of matrices as in definition 5. Let $\boldsymbol{\mu}^C = \{\boldsymbol{\mu}^C, C \in \mathcal{C}\}$ be a collection of vectors such that for each $B \subseteq C_1 \cap C_2$, $\boldsymbol{\mu}^{C_1}[B] = \boldsymbol{\mu}^{C_2}[B]$. Let κ and δ be non-negative real numbers. Since the marginal of a normal inverse Wishart depends only on the corresponding sub-vector and submatrix (see section B.3), the collection of laws

$$\mathcal{L}(\Sigma^C) = \mathcal{NIW}(\boldsymbol{\mu}^C, \kappa, \delta, \Phi^C)$$

is hyper consistent. The hyper Markov combination of its elements is called hyper normal inverse Wishart and noted $\mathcal{HNIW}(\boldsymbol{\mu}^C, \kappa, \delta, \Phi^C)$. By proposition 1 the \mathcal{HNIW} law is hyper Markov.

Theorem 2. *The \mathcal{HNIW} distribution is strong hyper Markov.*

Proof. Is a direct application of proposition 2. First, note that in our case, the set of models is the set of multidimensional Gaussian models

that factorize over \mathcal{G} . Thus, it is weak meta Markov and over a complete set, it forms a full exponential family. Since the \mathcal{HNIW} law is hyper Markov, and for any clique C , it is a conjugate distribution for the multidimensional Gaussian over C , by proposition 2 it is strong hyper Markov. \square

3.1.1 Conjugacy

Since the \mathcal{HNIW} is strong hyper Markov, we can update each of the marginal distributions locally. Furthermore, since the local laws are \mathcal{NIW} the update can be done using the result in appendix B.1. Hence, in order to learn from data we only have to update our hyperparameters using the equations (4)-(7).

3.1.2 Predictive distribution

The predictive distributions for each clique are multivariate t distributions, as given by equation (8). To get the distributions over a separator, we marginalize the distribution of one of the cliques that it separates using the result provided in section A.1 about the marginal of a multivariate t .

3.2 The Gaussian join tree classifier

As in CGN classifiers, Gaussian join tree classifiers start by determining a dependency structure and then adjust the model parameters for that structure (see Algorithm 1).

Algorithm 1 General GJT classifier

```

function GJTLARNER( $\mathcal{D}$ )
     $\mathcal{T} := \text{DetermineCliqueTree}(\mathcal{D})$ 
     $\Theta := \text{LearnGJTParameters}(\mathcal{T}, \mathcal{D})$ 
    return  $\langle \mathcal{T}, \Theta \rangle$ 
end function

```

In Gaussian join tree classifiers the structure is represented as a join tree. We formally define join trees following (Cowell et al., 1999).

Definition 7. A clique tree $\mathcal{T} = \{T_1, \dots, T_n\}$ is a tree where each node $T_i \subseteq X$ is a set of variables. If T_i is a parent of T_j , the separator between T_i and T_j is the set of variables $S_{ij} = T_i \cap T_j$. A join tree \mathcal{T} is a clique tree such that for every pair of nodes T_i, T_j , $T_i \cap T_j$ is a subset of every separator on the unique path from T_i to T_j .

The parameters of a GJT classifier are determined combining the results described in section 3.1.1 and 3.1.2. For each of the classes, each of the cliques and separators will be assigned a multivariate t distribution. The algorithmic details are provided in Algorithm 2.

3.3 Predicting

Given a new unclassified data point, Bayes formula is used to determine the posterior probability for each class. We use Laplace formula to assess the prior probability for class c (which is equivalent to assuming a Dirichlet prior) and get the posterior by multiplying it by joint probability obtained using equation (1).

4 Empirical comparison

In this section we compare GJT classifiers with classifiers making similar independency assumptions, but whose parameters are adjusted by maximum likelihood, namely CGN classifiers, as proposed in (Pérez, 2010). We use two different datasets from the bioinformatics domain, one for ovarian cancer and one for pancreatic cancer. Both datasets have been obtained from the NIH and contain high resolution spectrograms coming from surface-enhanced laser desorption/ionization time of flight mass spectrometry (SELDI-TOF MS). In both datasets the objective is to distinguish spectrograms coming from cancer patients from those coming from control individuals.

The ovarian cancer dataset contains a total of 216 spectrograms, 121 from cancer patients and 95 controls. The m/z values do not coincide along the different spectrograms. Thus, to create the variables, the m/z axis data has been discretized into different bins, creating a variable for each bin, for a total of 11300 variables. Thus, the number of variables largely exceeds the number of data points. For each spectrogram, the average of the values of each bin has been assigned to that bin's variable.

The pancreatic cancer dataset contains a total of 181 spectrograms, 101 from cancer patients and 80 controls. Each spectrogram is defined by 6771 variables which in this case are aligned, so no discretization is needed.

Algorithm 2 Bayesian learning of GJT parameters

```

1: function LEARNGJTPARAMETERS( $\mathcal{T}, \mathcal{D}$ )
2:   for each class  $c$  do
3:     Set the parameters of the prior distribution  $\kappa, \delta, \boldsymbol{\eta}$ , and  $\Psi$ .
4:     With the data from class  $c$  assess the number of data points  $n(c)$ , the sample mean  $\bar{\mathbf{y}}(c)$ 
5:     and the sample covariance matrix  $\Sigma(c)$ .
6:      $\kappa'(c) := \kappa(c) + n(c)$ 
7:      $\delta'(c) := \delta(c) + n(c)$ 
8:   end for
9:    $\mathcal{C} := \text{Cliques}(\mathcal{T})$ 
10:  for  $C \in \mathcal{C}$  do
11:    for each class  $c$  do
12:       $\boldsymbol{\eta}' := \frac{\kappa(c)\boldsymbol{\eta}[C] + n(c)\bar{\mathbf{x}}(c)[C]}{\kappa(c) + n(c)}$ 
13:       $\Psi' := \Psi[C, C] + (n(c) - 1)\Sigma(c)[C, C] + \frac{\kappa n(c)}{\kappa + n(c)}(\bar{\mathbf{x}}(c)[C] - \boldsymbol{\eta}[C])(\bar{\mathbf{x}}(c)[C] - \boldsymbol{\eta}[C])^T$ .
14:       $d(C, c) := t_{\delta'}(\boldsymbol{\eta}', \frac{\kappa' + 1}{\kappa' \delta'})\Psi'$ 
15:      for each clique  $C'$  child of  $C$  in  $\mathcal{T}$  do
16:         $d(C' \cap C, c) = t_{\delta'}(\boldsymbol{\eta}'[C' \cap C], \frac{\kappa' + 1}{\kappa' \delta'}\Psi'[C' \cap C, C' \cap C])$ 
17:      end for
18:    end for
19:  end for
20: end function

```

We have used two different families of structures. Both are based on the hypothesis that those variables that represent close m/z relations are more likely to have large correlations than those whose m/z values are further away.

A k -BOX structure can be defined over an ordered set of variables $V = \langle X_1, \dots, X_n \rangle$. It is a restriction of *semi Naïve Bayes* (Larrañaga et al., 2006), also known as *JAN* (Pérez, 2010). In a JAN structure the variables are joined in groups to form multivariate distributions. In the k -BOX structure we divide the variables in disjoint sets of k contiguous variables. The network structure can be seen in Figure 1(a) and the corresponding covariance matrix in Figure 2(a). In our case the ordering is provided by the m/z value.

The second proposed structure is the k -BAND structure. In k -BAND, we assume that each variable is independent of all the remaining variables given the $k - 1$ variables that precede it and the class variable. Therefore, the k -BAND structure is formed by cliques of size k with separators of $k - 1$ variables.

The covariance matrix for a k -BAND structure is a band of size k around the diagonal, as is shown in Figure 2(b). An example of the structure is shown in Figure 1(b).

We have run a sequence of experiments on

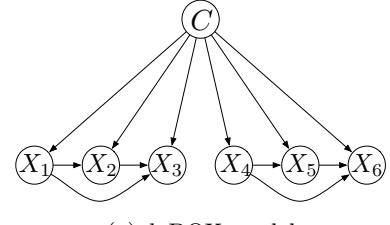
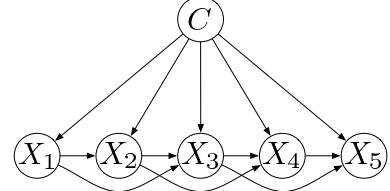
(a) k -BOX model(b) k -BAND model

Figure 1: Structure of the graph for different models

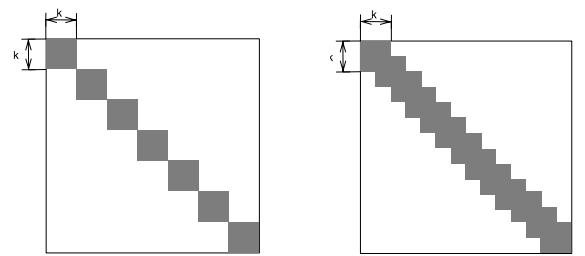


Figure 2: Structure of the covariance matrix

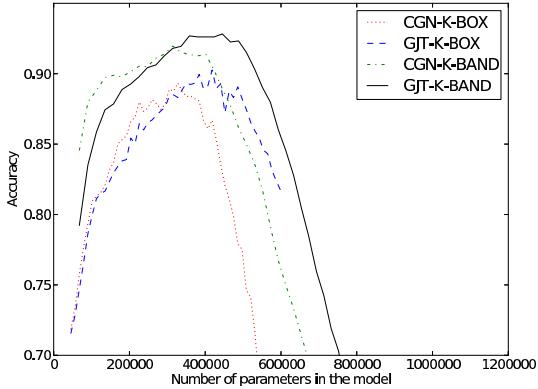


Figure 3: Prediction of ovarian cancer. Accuracy versus number of parameters in model.

each dataset to compare the different structures (k -BOX and k -BAND) and parameter learning methods (CGN and GJT) varying the k parameter.

We have run 5 repetitions of 10-fold cross validation and assessed the accuracy: the ratio of the number of data classified correctly to the total number of data classified; and conditional log-likelihood (CLL): the sum of the logarithm of the probability of the real class of the data given by the classifier. While accuracy gives us information about how many patients are correctly classified, CLL measures how accurately the probabilities for each class are estimated, which is very relevant for adequate decision making.

The ovarian data has been modelled using k -BOX and k -BAND structures with k ranging from 1 to 50. In Figure 3 we show the mean accuracy versus the number of parameters in the model and in Figure 4 we show the mean CLL versus the number of parameters in each structure. We see that in that dataset, k -BAND models are more accurate than k -BOX models. For low values of k , CGN performs better than GJT, but GJT has a largest accuracy and a highest CLL at its peak, and shows a more graceful decay as the number of parameters grows beyond that peak. The pancreatic cancer data has been classified using k -BOX and k -BAND structures with k ranging from 1 to 30. Figure 5 and Figure 6 show respectively

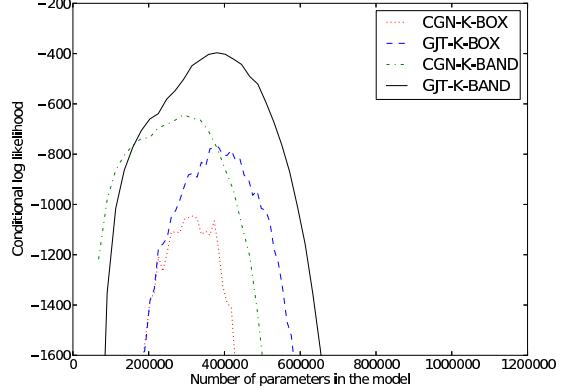


Figure 4: Prediction of ovarian cancer. CLL versus number of parameters in model.

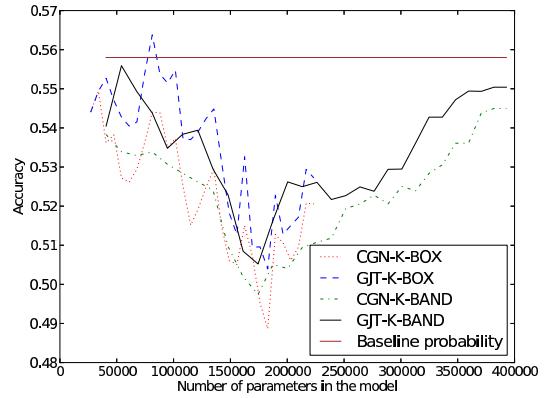


Figure 5: Prediction of pancreatic cancer. Accuracy versus number of parameters in model.

the mean accuracy and mean CLL against the number of parameters. Note that the accuracy for this dataset is much lower and close to the frequency of the largest class, that is 55.8% in this datasets. Previous studies have shown that the accuracy results for this dataset are much lower than for the previous one. The k -BOX model using GJT appears to reach the highest accuracy and the k -BAND model reaches the highest CLL also for pancreatic cancer.

The source code is available at <http://www.iiia.csic.es/~cerquide/pypermarkov>

5 Conclusions and future work

We have introduced a new family of classifiers for continuous domains, namely Gaussian join Tree classifiers that perform exact Bayesian av-

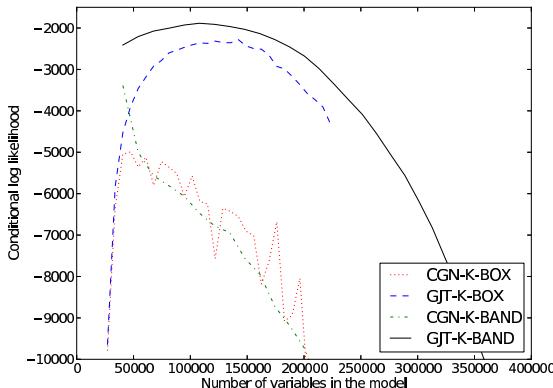


Figure 6: Prediction of pancreatic cancer. CLL versus number of parameters in model.

eraging over the parameters by virtue of the hyper normal inverse Wishart law, that we have introduced and proved to be strong hyper Markov. To assess the benefits we have compared GJT classifiers with CGN classifiers which assuming the same set of independencies adjust parameters using maximum likelihood. We performed our comparison with two high resolution mass spectrometry datasets for ovarian and pancreatic cancer prediction. We have seen that, for two simple dependency structures, our classifiers reach a better peak accuracy and a consistently better conditional log likelihood.

We have introduce k -BOX and k -BAND structure, which are part of the same family of join trees structures. For example different sizes of the separators define different models in the family. The study of this family remains also as future work.

The GJT parameter averaging can be performed over any set of dependencies that can be encoded into a decomposable graph. We can adapt any algorithm for learning the structure of a CGN classifier so that it outputs a join tree by moralizing and triangulating the DAG that encodes the structure of the CGN, and then running maximum cardinality search. A future line of work is comparing along the datasets in (Pérez, 2010) using the same structure learning algorithms that they suggest, thus testing if the benefits extend to domains with a smaller number of attributes.

Letac and Massam have generalized the hyper inverse Wishart distribution in (Letac and Massam, 2007). Studying the use of this generalized hyper inverse Wisharts for classification remains as future work.

Acknowledgments

We would like to thank Aritz Pérez, Steffen Lauritzen, Phil Dawid and Karina Gibert. This work has been funded by projects EVE (TIN2009-14702-C02-01 and TIN2009-14702-C02-02), CSIC 201050I008, and the Generalitat of Catalunya (2009-SGR-1434).

References

- Susanne Gammelgaard Böttcher. 2004. *Learning Bayesian Networks with Mixed Variables*. Ph.D. thesis, Aalborg University.
- Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer-Verlag.
- A. P. Dawid and S. L. Lauritzen. 1993. Hyper Markov Laws in the Statistical Analysis of Decomposable. *The Annals of Statistics*, 21(3):1272–1317.
- Dan Geiger and David Heckerman. 1994. Learning gaussian networks. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 235–243.
- Andrew Gelman, JB Carlin, HS Stern, and Rubin. 2004. *Bayesian data analysis*.
- Samuel Kotz and Saralees Nadarajah. 2004. *Multivariate T-Distributions and Their Applications*. Cambridge University Press, Cambridge.
- Pedro Larrañaga, P, Aritz Pérez, Iñaki Inza, and Pedro Larra. 2006. Supervised classification with conditional Gaussian networks : Increasing the structure complexity from naive Bayes. *International Journal of Approximate Reasoning*, 43(January):1–25.
- Gérard Letac and Hélène Massam. 2007. Wishart distributions for decomposable graphs. *The Annals of Statistics*, 35(3):1278–1323, July.
- Aritz Pérez. 2010. *Supervised classification in continuous domains with Bayesian networks*. Ph.D. thesis, Universidad del País Vasco.

A Multivariate t-distributions

A p -dimensional random vector \mathbf{x} is said to have the p -variate t distribution with degrees of freedom ν , mean vector $\boldsymbol{\mu}$, and correlation matrix R (that is $x \sim t_\nu(\boldsymbol{\mu}, R)$) if its joint pdf is given by

$$p(\mathbf{x}|\nu, \boldsymbol{\mu}, R) = \frac{\Gamma((\nu+p)/2)}{(\pi\nu)^{p/2}\Gamma(\nu/2)|R|^{1/2}} \cdot [1 + \frac{1}{\nu}(x - \boldsymbol{\mu})R^{-1}(x - \boldsymbol{\mu})]^{-(\nu+p)/2} \quad (2)$$

A.1 Marginals

Let \mathbf{x} be a p -dimensional random vector $\mathbf{x} \sim t_\nu(\boldsymbol{\mu}, R)$. Furthermore let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ where \mathbf{x}_1 is p_1 dimensional, $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)$ and $R = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix}$. We have that

$$\mathbf{x}_1 \sim t_\nu(\boldsymbol{\mu}_1, R_{11}) \quad (3)$$

For more information regarding multivariate t -distributions see (Kotz and Nadarajah, 2004).

B Normal Inverse Wishart distributions

The inverse Wishart distribution is defined on real-valued positive-definite $p \times p$ matrices. It has two parameters: a real number $\delta > 0$ and a $p \times p$ positive-definite matrix Ψ . The probability density function is

$$\mathcal{IW}(X|\delta, \Psi) = \frac{|\Psi|^{\frac{\delta+p-1}{2}} |X|^{-\frac{\delta+2p}{2}}}{2^{\frac{(\delta+p-1)p}{2}} \Gamma_p(\frac{\delta+p-1}{2})} e^{-\frac{1}{2}tr(\Psi X^{-1})}$$

The normal inverse Wishart distribution is defined on pairs composed of (i) vectors of dimension p and (ii) real-valued positive-definite $p \times p$ matrices. It has four parameters: a p dimensional vector $\boldsymbol{\eta}$ that encodes the location, a positive real number κ that acts as scaling factor, and δ and Ψ as in the inverse Wishart. The probability density function is

$$\mathcal{NIW}(\boldsymbol{\mu}, \Sigma|\boldsymbol{\eta}, \kappa, \delta, \Psi) = \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\eta}, \frac{1}{\kappa}\Sigma) \cdot \mathcal{IW}(\Sigma|\delta, \Psi)$$

B.1 Conjugacy

The normal inverse Wishart distribution is conjugate to the multivariate normal (Gelman et al., 2004). Thus, if we assume as prior a

$\mathcal{NIW}(\boldsymbol{\mu}, \Sigma|\eta, \kappa, \delta, \Psi)$ and we are given a sample X from a multivariate normal, the posterior will be a $\mathcal{NIW}(\boldsymbol{\mu}, \Sigma|\eta', \kappa', \delta', \Psi')$ where

$$\boldsymbol{\eta}' = \frac{\kappa\boldsymbol{\eta} + n\bar{\mathbf{x}}}{\kappa + n}, \quad (4)$$

$$\kappa' = \kappa + n, \quad (5)$$

$$\delta' = \delta + n, \quad (6)$$

$$\Psi' = \Psi + (n-1)S + \frac{\kappa n}{\kappa + n}(\bar{\mathbf{x}} - \boldsymbol{\eta})(\bar{\mathbf{x}} - \boldsymbol{\eta})^T. \quad (7)$$

and S is the sample covariance.

B.2 Predictive distribution

The predictive distribution is

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\eta}, \kappa, \delta, \Psi) &= \\ &= \int_{\boldsymbol{\mu}, \Sigma} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) \mathcal{NIW}(\boldsymbol{\mu}, \Sigma|\boldsymbol{\eta}, \kappa, \delta, \Psi) = \\ &= t_\delta(\boldsymbol{\eta}, \frac{\kappa+1}{\kappa\delta}\Psi). \end{aligned} \quad (8)$$

B.3 Marginals

Proposition 3. Let $\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}$, $\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$, $\boldsymbol{\eta} = \begin{pmatrix} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \end{pmatrix}$ and $\Psi = \begin{pmatrix} \Psi_{11} & \Psi_{12} \\ \Psi_{21} & \Psi_{22} \end{pmatrix}$. If $(\boldsymbol{\mu}, \Sigma) \sim \mathcal{NIW}(\boldsymbol{\eta}, \kappa, \nu, \Psi)$, then $(\boldsymbol{\mu}_1, \Sigma_{11}) \sim \mathcal{NIW}(\boldsymbol{\eta}_1, \kappa, \nu, \Psi_{11})$

Proof. The marginal can be assessed as follows

$$\begin{aligned} P(\boldsymbol{\mu}_1, \Sigma_{11}|\boldsymbol{\eta}, \kappa, \delta, \Psi) &= \\ &= \int_{\boldsymbol{\mu}_2, \Sigma_{12}, \Sigma_{22}} \mathcal{NIW}(\boldsymbol{\mu}, \Sigma|\boldsymbol{\eta}, \kappa, \delta, \Psi) = \\ &= \int_{\boldsymbol{\mu}_2, \Sigma_{12}, \Sigma_{22}} \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\eta}, \frac{1}{\kappa}\Sigma) \cdot \mathcal{IW}(\Sigma|\delta, \Psi) = \\ &= \int_{\boldsymbol{\mu}_2, \Sigma_{12}, \Sigma_{22}} \mathcal{N}(\boldsymbol{\mu}_1|\boldsymbol{\eta}_1, \frac{1}{\kappa}\Sigma_{11}) P(\boldsymbol{\mu}_2|\boldsymbol{\mu}_1, \boldsymbol{\eta}, \kappa, \delta, \Psi) \cdot \\ &\quad \cdot \mathcal{IW}(\Sigma_{11}|\delta, \Psi_{11}) P(\Sigma_{22}, \Sigma_{21}|\Sigma_{11}, \delta, \Psi) = \\ &= \mathcal{N}(\boldsymbol{\mu}_1|\boldsymbol{\eta}_1, \frac{1}{\kappa}\Sigma_{11}) \mathcal{IW}(\Sigma_{11}|\delta, \Psi_{11}). \end{aligned} \quad (9)$$

□

Interactive learning of Bayesian Networks using OpenMarkov

Iñigo Bermejo

UNED, Spain

ibermejo@dia.uned.es

Jesús Oliva

CSIC, Spain

jesus.oliva@car.upm-csic.es

Francisco Javier Díez

UNED, Spain

fjdiez@dia.uned.es

Manuel Arias

UNED, Spain

marias@dia.uned.es

Abstract

Algorithms for learning Bayesian networks (BNs) behave as a black box that takes a database as an input and returns a network as the output. In contrast, OpenMarkov, our tool for probabilistic graphical models, includes the option to run the algorithms in a step-by-step fashion, presenting a ranked list of operations (such as adding, removing, or inverting links) the user can select, while allowing live edition of the BN throughout the learning process. The application offers some data preprocessing options and the possibility to use a model network to guide the learning process. This functionality in OpenMarkov can be employed to learn BNs with partial expert knowledge, to debug new algorithms, and as a pedagogical tool.

1 Introduction

A probabilistic graphical model (PGM) consists of a joint probability distribution defined on a set of variables \mathbf{V} and a graph containing a node for each variable X in \mathbf{V} ; the structure of the graph imposes some relations of conditional independence on the structure of the network, which depend mainly on the type of graph. Some types of PGMs are Bayesian networks (BNs), Markov networks, influence diagrams, hidden Markov models, factored MDPs and POMDPs, etc.¹

In many cases PGMs are built from expert knowledge: causal relations are used to draw the arcs of the graph, and the conditional prob-

abilities are obtained from the literature (for example, medical journals), from databases, or from experts' estimations. The difficulty and tediousness of this approach has led to an increasing interest for learning methods that can generate PGMs from databases automatically. This is the preferred approach when there is a large database with few or none missing values, and no causal knowledge. However, in many cases the size of the database does not allow to learn a PGM that accurately represents the conditional independencies existing in the domain of application. Practitioners of these methods often encounter that the PGM obtained contains some links that the expert considers as obviously spurious, but given that in general the algorithms perform as black boxes, it is diffi-

¹www.cisiad.uned.es.

cult to determine to what degree those links are really supported by the data.

Another problem is that most learning algorithms do not return causal models. In the last decade the number of studies aimed at obtaining causal models from databases has grown exponentially (the UAI Conference held in Barcelona in July 2011 was a clear illustration of this phenomenon). However in many cases the problem does not lie in the algorithm but on the lack of information in the database: the only conclusion that can be drawn reliably from a set of data—provided that it is big enough and not biased—is the set of correlations that exist in the real world. These correlations rule out some causal models, but the number of models compatible with the data is usually very large. Many of those models clash with common knowledge of the experts, but it is not easy to feed that knowledge into automatic causal learning algorithms. For this reason, it would be useful to have interactive learning algorithms that propose a list of changes to improve the accuracy of the network but allow an expert to select only those that do not contradict his/her knowledge.

Secondly, interactive learning could be also of great interest for researchers and developers of new algorithms. An interactive learning tool able to show on a graphical interface the different actions that the algorithm is considering at each step and the scores assigned to them may be very useful to debug the algorithm, for example, by observing how a shift in some of the parameters leads to a different selection of actions.

Thirdly, interactive learning programs may have a high pedagogical value by allowing the students to know the actions that the algorithm has evaluated at each step and why it has selected each action. Then it is possible to run a different algorithm, for example with a different search strategy or a different metric, and observe why it selects different actions at each step.

For these reasons we decided to implement an interactive learning module on **OpenMarkov**, an open-source software tool for editing and eval-

uating PGMs. The interactive learning module includes a user-friendly graphical interface that allows to overcome all the above-pointed problems, with the corresponding benefit for experts, researchers, developers and students.

The rest of this paper is structured as follows. In Section 2 we give a brief overview of Bayesian network learning and of the **OpenMarkov** tool. Section 3 describes the different options available to the user for learning BNs interactively in **OpenMarkov**. Section 4 presents a case study: how to learn interactively the BN *Alarm*, a model frequently used in the literature as a benchmark for learning algorithms. In Section 5 we discuss the advantages of our approach and similar approaches, and Section 6 contains the conclusions and proposals for future work.

2 Background

2.1 Learning Bayesian networks

Learning Bayesian networks is one of the most important research areas in the field of BNs. Every year, around one third of the total publications in that area are related to automatic learning. Just like in manual construction, automatic learning of BNs presents two aspects: parametric learning and structural learning. Parametric learning consists of computing the conditional probabilities given by the structure of the network using the observed frequencies on the database. Structural learning tries to find the graph that best represents the probability distribution based on the frequencies in the database.

Structural learning methods There are two main methods for building the graph of a BN from a database. The first one consist of detecting the probabilistic conditional independencies present in the database. The most famous algorithm of this type is the *PC algorithm* (Spirtes and Glymour, 1991; Spirtes et al., 2000). The second method, called *search and score*, consists of performing a heuristic search through the space of possible structures, using a metric that measures how well each structure can represent the probability distribution of the variables in the database. Several

metrics have been proposed in the literature: Bayesian (which include K2 and BDe as particular cases), cross-entropy, AIC, and MDL—see (Bouckaert, 2004) for references. K2, the first algorithm of this type, performed a search by departing from a network without links and adding at each step the link leading to the highest score, provided that the score was positive (Cooper and Herskovits, 1991). The method that proceeds by examining one operation at each step (adding, removing, or inverting a link) is called *hill climbing*.

2.2 OpenMarkov

This project started in 2002 at the Department of Artificial Intelligence of the Universidad Nacional de Educación a Distancia (UNED), in Madrid, Spain. Its original name was Carmen (Arias and Díez, 2008), but in 2010 it was renamed as OpenMarkov.² We departed from our experience in the construction of Elvira (Elvira Consortium, 2002),³ an open-source tool begun in 1997 as a joint project of several Spanish universities, but everything in the new program was redesigned and the code of OpenMarkov was built from scratch. The language chosen to develop OpenMarkov was Java, mainly to make it multi-platform.

OpenMarkov is able to represent several types of networks, such as Bayesian networks, Markov networks, influence diagrams, LIMIDs, and decision analysis networks (DANs), as well as several types of temporal models: dynamic Bayesian networks, Markov processes with atemporal decisions (MPADs), MDPs, POMDPs, Dec-POMDPs, and dynamic LIMIDs—see (Arias et al., 2011) for definitions and references. Currently it can only evaluate Bayesian networks, influence diagrams, and MPADs. Each network type is defined by a set of constraints (Arias et al., 2011, Appendix A), which leads to the possibility of defining new types of networks easily by combining the existing constraints and, if necessary, by adding new ones. Constraints play also an important

role in the learning of BNs, as we will discuss below.

There are three types of variables in OpenMarkov: finite-states, numerical, and discretized. A discretized variable has a finite set of states, each one having an associated numeric interval.

The graphical user interface (GUI) is very similar to those of other software tools for PGMs, especially to that of Elvira. It has two working modes: edition and inference. It has been designed for internationalization; currently messages can be displayed in English and Spanish. For further details, see OpenMarkov's web pages and wiki.⁴

3 Options for learning BNs in OpenMarkov

In this section we describe the main options that OpenMarkov offers for learning BNs interactively.

3.1 Using a model network

OpenMarkov gives the user the option to use an existing network as a model for the one that will be learned. There are four options. The first is to use the model only to determine the positions of the nodes. When we learn a network from a database we can place the nodes on the screen by dragging them with the mouse, trying to minimize the number of links crossing one another; but if we learn another network from the same database (for example, using different options for the algorithm), we should drag and replace the nodes again. OpenMarkov facilitates this task by placing the nodes in the same positions as in a network built previously.

The other three options are whether the algorithm can add, remove, or invert the links present in the model network. They are useful, for example, when we wish that the algorithm preserves all the links in the model network. The first option (i.e., using the model only to place the nodes) is incompatible with the other three, which are compatible with one another.

²www.openmarkov.org.

³www.ia.uned.es/~elvira.

⁴www.openmarkov.org, wiki.openmarkov.org.

There are other uses of the model network, that we describe below.

3.2 Data preprocessing

Unfortunately data in the databases is usually not suitable to be directly fed to the learning algorithm and has to be preprocessed. OpenMarkov offers the following options.

Selection of variables Usually raw databases contain information that is irrelevant for the model (e.g. the patient’s name). OpenMarkov can learn a network that contains all the variables in the database, but it is also possible to tell it to use only those present in the model network. The third possibility is to select the variables one by one from a list.

Discretization of numeric variables Currently OpenMarkov can only learn BNs with variables having finite states. Therefore, the numeric variables in the database must be discretized before feeding the data to the learning algorithm. OpenMarkov can discretize a variable in different ways. First, the user can indicate a number of intervals and then indicate whether the intervals must have the same width (considering the maximum and the minimum for that variable in the database) or the same frequency (i.e., the number of database registers for every interval will be the same). Second, if the variable is discretized in the model network, its intervals can be used to assign each number in the database to a state. For example, if a variable has three states, “negative”, “null”, and “positive”, with three associated intervals, $(-\infty, 0]$, $[0, 0]$, and $(0, +\infty)$, respectively, these intervals can be used to discretize the values in the database. This way, creating a model network is a way of specifying how numeric variables should be discretized.

Imputation of missing values Currently OpenMarkov offers only two ways to fill in the gaps in the database: either to ignore every register that contains at least one missing value, or to write the value “missing” in every empty cell, which is then treated as if it were an ordinary value.

3.3 List of suggested edits

In OpenMarkov an *edit* is an atomic modification of a data structure. There are three edits that an interactive learning algorithm can propose: adding, removing, or inverting a link. The list is composed by sorting the edits according to their scores. The hill climbing algorithm computes the scores using the metric selected by the user. The PC algorithm performs many statistical test in which the null hypothesis is that two variables are not correlated given other variables. Roughly speaking, a high p resulting from the test suggests that two variables are conditionally independent, i.e., that a link can be removed. Therefore, the p value can be used as a score to rank the edits, each edit being the removal of a link.

Interactive learning is performed by having two windows: one showing the graph of the network and another one showing the proposed edits. The user can select any edit from the list, not necessarily the one having the highest score, and the change will be immediately displayed on the network window. Alternatively, the user can add or remove any link from the graph. In both cases, the scores will be recalculated and a new list will be proposed. Figure 1 shows the lists of suggested edits shown during the interactive learning process.

Additional options There are additional options to control the flow of the algorithm. One of them is tell OpenMarkov to show only the edits having a positive score. Another option is to show only the edits allowed by the constraints associated to the network. For example, a constraint stemming from the definition of the BN is that the graph cannot contain cycles. A constraint that the user can impose is that a node cannot have more than n parents. If the user selects the “Show only allowed edits” option, those incompatible with the constraints will not be shown in the list, even if they have a high score.

Finally, the user has the possibility of blocking a certain edit to prevent the system from offering it again and again. Blocked edits can be later unblocked at any moment.

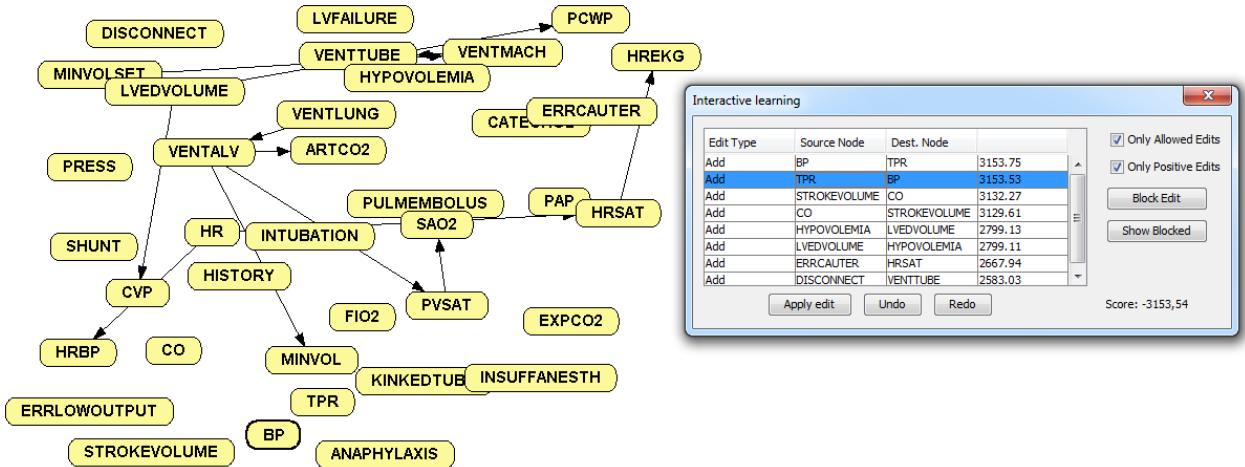


Figure 1: A moment of the interactive learning process: list of edits proposed by OpenMarkov and the network being learned.

4 Case study

In order to explore the benefits of interactive learning, we study the case of learning the well-known ALARM network (Beinlich et al., 1989), which has 37 nodes and 46 links. The nodes are classified into three levels. The first level contains *diagnostic nodes*, which have no predecessors. The second level contains *intermediate variables*, representing pathophysiological anomalies that cannot be observed directly. The third contains *measurement nodes*, which represent clinical variables that can be observed or measured, and do not have children. The network contains no link from a lower level to an upper level.

Using this network we generated a database containing 10,000 samples and applied the hill-climbing algorithm with the K2 metric. We applied the learning algorithm automatically in OpenMarkov, resulting in a model with 50 links, 13 of which were not in the original network, even though 6 of them were inverted links of the original network. On the other hand, 9 of the original links were missing in the network learned.⁵

Then we learned the network interactively using elementary causal knowledge, according to

⁵The files used in this study are available at www.openmarkov.org/learning so that the results can be reproduced by other researchers.

which we did not accept the addition of any link from a measurement node to an intermediate or a diagnostic node, nor from an intermediate node to a diagnostic node. Figure 2 shows an example of a moment in the interactive learning process where the edit with the highest score contravenes the causal knowledge. In this case, we chose to apply the second edit, which produces the same link but in the opposite direction.

The resulting net contained 47 links: only 2 of them were not in the original network and only one of the original links was missing. The two links “invented” by the learning algorithm were the last to be added and had such a low score that they might have been detected as spurious by the expert. We also observed that the missing link, from INSUFFANESTH to CATECHOL, has a very weak influence in the original network.

This experiment shows that even a portion of very rudimentary knowledge about the domain may lead to a significant improvement in the network built by our interactive learning algorithm.

5 Discussion

5.1 Advantages of our approach

As mentioned in the introduction, a problem of learning algorithms is that they often create spurious links due to small correlations exist-

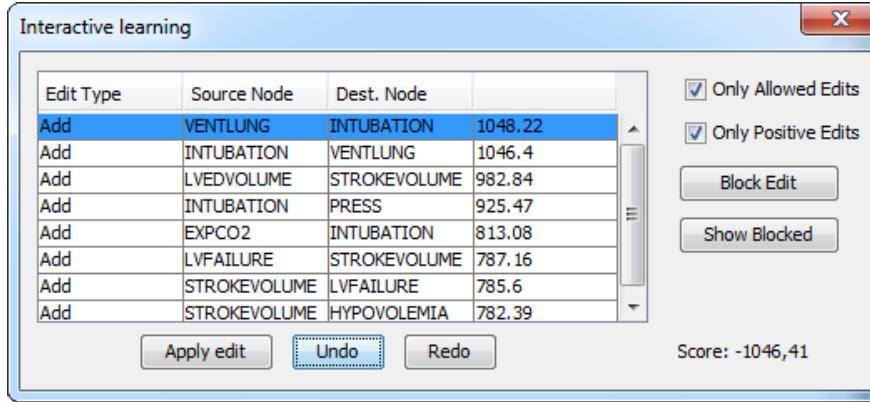


Figure 2: The edit suggested at the top of the list contravenes our causal knowledge because VENTLUNG is an intermediate variable and INTUBATION is a diagnostic variable.

ing in the database. Another problem is that in general the models obtained are not causal, not only because of the inherent limitation of most algorithms, but mainly because the information contained in the database does not permit to distinguish whether X is a direct cause of Y , or X is a cause of Y , or if there is a directed causal path between them involving other variables, or they have a common cause, or there is a selection bias in the database (Glymour and Cooper, 1999; Druzdzel and Díez, 2003).

OpenMarkov allows human users, who may be experts in their respective areas but novices in the field of probabilistic modelling, to supervise the execution of learning algorithms. The algorithm proposes some incremental modifications of the network, based on the information contained in the database, and the user has the opportunity to apply some of the changes proposed by the tool or impose others at any moment of the learning process, based on their expertise. Even if this might lead to a lower quality of the network according to the metric, the result might be better from the point of view of users' acceptability, because human experts are reluctant to accept the advice of a machine if they cannot follow its reasoning (Teach and Shortliffe, 1984).

An interactive learning tool might as well be useful for researchers that have developed a new algorithm and wish to trace its execution in order to debug or fine-tune it. This process can

be done by inserting in the algorithm a few lines of code that print a trace on the standard output or in a file, but it is much nicer to observe graphically the operations performed by the algorithm, step by step, together with the qualitative information associated with the next modifications that the algorithm has evaluated. Obviously, this requires that the researchers implement the new functionality (such as a new metric, a new search technique, or a completely novel learning method). OpenMarkov's architecture has been carefully designed to permit these extensions: each new method can be implemented as a Maven subproject, that OpenMarkov will detect at run time as a plug-in.⁶ This way, researchers can extend OpenMarkov dynamically without modifying the “official” source code.

Finally, an interactive learning program may be useful as a pedagogical tool to explain the performance of different algorithms: rather than observing the input and the output, students may follow the progression of the algorithm step by step, understanding why each change was selected, seeing the effects of taking different actions to those proposed by the system and comparing different algorithms.

5.2 Related work

Interactive structural learning was proposed by Sucar and Martínez-Arroyo (1998) as a means

⁶See <https://bitbucket.org/cisiad/org-openmarkov/wiki/OpenMarkov-organization.pdf>.

for human-computer collaboration in the search for the optimal network structure. Their system initially builds a tree automatically and then allows the user to modify it by adding, removing or inverting arcs. The strength of the correlation between the variables connected by a link is denoted graphically by the width of the link. It also shows a score representing the quality of the model, which is inversely related with the complexity of the network and the distance between the probability distributions specified by the network and the data.

Our work, developed independently, is also based on the idea of combining a learning algorithm and expert knowledge to build a model interactively. The main difference is that our system guides the expert through the creation process step-by-step instead of providing the final result of the learning algorithm and asking the expert to fine tune it. We think that our approach contributes to a better understanding of the behavior of the learning algorithm and therefore makes collaboration with the expert easier.

The idea of showing a score that represents the quality of the model is also present somehow in our approach, because the scores associated to each edit in the score and search algorithm represent the increment in the quality of the model, given by the complexity of the network and the distance between the probability distribution of the network and that of the data.

In the future, we will implement in **OpenMarkov** the possibility of representing the type of correlation (positive, negative, or null) by a color and the strength of the correlation by the thickness of the link, as we did in Elvira (Lacave et al., 2006; Lacave et al., 2007), thus making our approach more similar to that of Sucar and Martínez-Arroyo.

Later de Campos and Castellano (2007) encoded expert knowledge in the form of a set of restrictions that the learning algorithm had to satisfy. They used three types of restrictions: presence of a link, absence of a link, and partial ordering of the variables. They proved that, for several data sets, this approach improved the quality of the networks learnt. Our work is sim-

ilar in that, by means of the model network, we can impose or prevent the presence of some links in the network learnt, but we do not have yet partial order restrictions. However, it would be easy to implement any restriction—not only those used by de Campos and Castellano—as an **OpenMarkov** constraint, as explained in Section 3.3. Another difference is that in **OpenMarkov** it is possible to specify whether the model network imposes the presence or absence of a link, or it only “suggests” them as a starting point, that can be overridden depending the scores computed by the algorithm.

Another similarity between their work and ours is that both of them have been combined with score-and-search and independence-based algorithms. However, the main difference is that their approach is not interactive: their restrictions must be declared before the execution of the algorithm, which then runs automatically, while in **OpenMarkov** every action proposed by the algorithm can be accepted or rejected by the user, who can also impose any action at any moment of the process, thus giving full control to the user.

6 Conclusions and Future Work

In this paper we have described an interactive learning approach for learning Bayesian networks from databases, which may be very useful for the experts in different application domains, as well as for researchers and students in the field of PGMs. We have shown with a case study that even very rudimentary causal knowledge about the domain may lead to a significant improvement of the network build interactively with a learning algorithm.

The main lines for future development would be to borrow some ideas from the work of Sucar and Martínez-Arroyo (1998) and de Campos and Castellano (2007), such as representing graphically the strength of the correlation between variables and having richer types of constraints. It would be also useful to show an absolute quality measure of the net rather than the incremental one we currently have, given by the complexity of the network and the

distance between the probability distribution of the network and that of the data. This quality measure could be used to compare the resulting nets of the interactive and non-interactive learning processes.

Another research line would be to adapt our approach to learning Bayesian classifiers, a somewhat different problem, as the objective is not to build the network that better represents the probability distribution of the data, but the network that better classifies new cases.

Acknowledgments

We thank Concha Bielza, Pedro Larrañaga and the reviewers for their useful comments. This work has been supported by grants TIN2006-11152 and TIN2009-09158, from the Spanish Ministry of Science and Technology. I.B. has received a predoctoral fellowship from the Universidad Nacional de Educación a Distancia (UNED), and J.O. from the Consejo Superior de Investigaciones Científicas (CSIC).

References

- [Arias and Díez2008] M. Arias and F. J. Díez. 2008. Carmen: An open source project for probabilistic graphical models. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM'08)*, pages 25–32, Hirtshals, Denmark.
- [Arias et al.2011] M. Arias, F. J. Díez, and M. P. Palacios. 2011. ProbModelXML. A format for encoding probabilistic graphical models. Technical Report CISIAD-11-02, UNED, Madrid, Spain.
- [Beinlich et al.1989] I. A. Beinlich, H. J. Suermondt, R. M. Chávez, and G. F. Cooper. 1989. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the 2nd European Conference on AI and Medicine*, pages 247–256, London. Springer-Verlag, Berlin.
- [Bouckaert2004] R. R. Bouckaert. 2004. Bayesian networks in Weka. Technical Report 14/2004, Computer Science Department, University of Waikato, New Zealand.
- [Cooper and Herskovits1991] G. F. Cooper and E. Herskovits. 1991. A Bayesian method for constructing Bayesian belief networks from databases. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence (UAI'91)*, pages 86–94, Los Angeles, CA. Morgan Kaufmann, San Mateo, CA.
- [de Campos and Castellano2007] L. M. de Campos and J. G. Castellano. 2007. Bayesian network learning algorithms using structural restrictions. *International Journal of Approximate Reasoning*, 45:233–254.
- [Druzdzel and Díez2003] M. J. Druzdzel and F. J. Díez. 2003. Combining knowledge from different sources in probabilistic models. *Journal of Machine Learning Research*, 4:295–316.
- [Elvira Consortium2002] The Elvira Consortium. 2002. Elvira: An environment for creating and using probabilistic graphical models. In J. A. Gámez and A. Salmerón, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM'02)*, pages 1–11, Cuenca, Spain.
- [Glymour and Cooper1999] C. Glymour and G. F. Cooper. 1999. *Computation, causation and discovery*. The MIT Press, Cambridge, Massachusetts.
- [Lacave et al.2006] C. Lacave, A. Onisko, and F. J. Díez. 2006. Use of Elvira’s explanation facilities for debugging probabilistic expert systems. *Knowledge-Based Systems*, 19:730–738.
- [Lacave et al.2007] C. Lacave, M. Luque, and F. J. Díez. 2007. Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics*, 37:952–965.
- [Spirtes and Glymour1991] P. Spirtes and C. Glymour. 1991. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62–72.
- [Spirtes et al.2000] P. Spirtes, C. Glymour, and R. Scheines. 2000. *Causation, Prediction and Search*. The MIT Press, Cambridge, Massachusetts, second edition.
- [Sucar and Martínez-Arroyo1998] L. E. Sucar and M. Martínez-Arroyo. 1998. Interactive structural learning of bayesian networks. *Expert Systems with Applications*, 15:325–332.
- [Teach and Shortliffe1984] R. L. Teach and E. H. Shortliffe. 1984. An analysis of physician’s attitudes. In B. G. Buchanan and E. H. Shortliffe, editors, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, chapter 34, pages 635–652. Addison-Wesley, Reading, MA.

Tools and Algorithms for Causally Interpreting Directed Edges in Maximal Ancestral Graphs

Giorgos Borboudakis, Sofia Triantafillou, Ioannis Tsamardinos
Computer Science Department, University of Crete, Greece
Institute of Computer Science, FORTH, Greece
borbudak@ics.forth.gr

Abstract

The Maximal Ancestral Graph (MAG) formalism is an important generalization of Bayesian Networks for representing causal processes that admit the possibility of latent confounding variables. Thus, when learning MAGs from data for Causal Discovery, the often unrealistic assumption of Causal Sufficiency can be dismissed. However, the *causal interpretation* of edges in a MAG is not trivial and it is potentially misleading to unfamiliar practitioners. An edge $X \rightarrow Y$ may denote either (a) X causes Y and no latent confounding variable is present (*pure-causal* edge) or (b) X causes Y with the potential presence of a latent common cause. In addition, an edge $X \rightarrow Y$ may denote (I) X causes Y directly (*direct-causal* edge), i.e., without any modeled variables mediating the causation or (II) X causes Y possibly-indirectly. In this paper, we present polynomial-time algorithms and tools that can distinguish among the above cases and facilitate the causal interpretation of MAGs. In addition, we run simulated experiments to quantify the percentage of edges that can be labeled as pure or direct-causal. Our results show that the percentage of edges that can be labeled as pure-causal achieves a minimum for sparse or dense networks, and a maximum for in-between values of edge density. In contrast, the percentage of edges that can be labeled as direct-causal decreases as the edge density of the MAG increases.

1 Introduction

A Causal Bayesian Network (CBN) (Pearl, 2000) is a probabilistic graphical model that can represent a data distribution. The graph of a CBN encodes a set of conditional independencies on the observed variables provided by the d -separation criterion. In addition, an edge $X \rightarrow Y$ in the graph has *causal* semantics: [X *directly causes* Y], that is an intervention of X has a direct effect on Y . The meaning of “direct” causation is that the causation persists even when all other *observed* variables are held constant, for some appropriate combination of values of the other variables (Spirtes et al., 2000). Thus, direct causation depends on the context of observed variables.

A major assumption of CBNs is that there are no latent confounding variables, i.e., latent common causes of the observed variables,

named the *Causal Sufficiency assumption*. If this is not the case, then a system may not be representable so that both the probabilistic *and* the causal semantics are correct. For example, let us assume that the true causal structure is $X \leftarrow H \rightarrow Y$, H is latent, and X and Y are dependent. There are three possible CBNs. The graph $[X \text{ (no edge)} Y]$, entails X is independent of Y and does not comply with the probabilistic semantics. The graphs $[X \leftarrow Y]$ and $[X \rightarrow Y]$ correctly represent the distribution but do not comply with the causal semantics.

The Maximal Ancestral Graph (MAG) formalism (Richardson and Spirtes, 2002) is an extension of CBNs which admits the presence of hidden confounders, without explicitly introducing them in the model. An additional important feature of MAGs is that they are closed under marginalization; given a MAG M , the

MAG M' representing the causal relations after marginalizing out a set of variables can be computed. The probabilistic interpretation of the edges also stems from a simple generalization of the d -separation criterion: the m -separation. The semantics of an edge $X \leftrightarrow Y$ are that [neither X causes Y nor the reverse]. An edge $X \leftrightarrow Y$ may be required in the model to represent possible latent confounders: $X \leftarrow H \rightarrow Y$ can be encoded as the MAG $X \leftrightarrow Y$.

However, the advantages of MAGs come at the price of *difficulty in the causal interpretation of the directed edges*. The causal semantics of a directed edge $X \rightarrow Y$ are that [X causes Y]. The causal relation may be **direct-causal** (no observed variables mediate it) or it may be **possibly-indirect** and mediated by the *observed* variables. In addition, the edge $X \rightarrow Y$ does not distinguish between two possible scenarios: (a) X causes Y but there also is a latent confounder H of the two variables (thus, the graph is $[X \rightarrow Y \leftarrow H \rightarrow X]$), called a **possibly-confounded** edge, and (b) X causes Y and there are no latent confounders. We call the later a **pure-causal** edge (a *visible edge* in the terminology of (Zhang, 2008)).

In this paper, based on results from (Zhang, 2008), we present algorithms that identify all the direct-causal and pure-causal directed edges in a MAG. The algorithms are complete in the sense that if an edge is labeled as possibly-indirect then it could be the marginal of a DAG where it is mediated by observed variables; similarly, if it is labeled as possibly-confounded it could be the marginal of a DAG containing a marginalized common cause of the edge-points.

We consider labeling **pure-causal** and **direct-causal** edges important for interpreting the graph. For a confounded edge $X \rightarrow Y$ the correlation (equivalently dependency, mutual information) between X and Y is partly due both to the direct causal effect as well as due to the latent confounder (as well as other paths). When $X \rightarrow Y$ is pure-causal edges there are no latent confounders to contribute to the correlation, which makes identification of the total or direct effect, easier (see (Cai et al., 2008) for more details). Similarly, a direct-causal edge

$X \rightarrow Y$ implies that the causal effect of X persists even when all other observed variables are held fixed (for some appropriate set of their values).

Figure 1 shows an example encompassing all concepts and cases of this paper. Figure 1(c) shows a MAG annotated by our tools. Directed edges are partitioned into four categories for the four combinations of pure or possibly-confounded causal edge, and direct or possibly-indirect causal edge. Figure 1(a) and Figure 1(b) show two DAGs that could produce the MAG after marginalizing the H variables. Finally, in simulated experiments we show that the percentage of edges that can be labeled as pure-causal or direct-causal depends on the density of the graph.

2 Background

We briefly review the Maximal Ancestral Graph (MAG) formalism (Richardson and Spirtes, 2002) and basic background concepts. MAGs can deal with selection variables but this is out of the scope of this paper (i.e., we actually consider the sub-class called *Directed Maximal Ancestral Graphs* (DMAGs)).

We denote a variable with an upper-case letter (e.g. X) and a set of variables with upper-case bold letters (e.g. \mathbf{Z}). A *directed mixed graph* $G = (V, E)$ consists of a set of vertices V and a set of edges E . We denote with $n = |V|$ the number of vertices and with $m = |E|$ the number of edges of G . Directed mixed graphs contain two kinds of edges: directed edges (\rightarrow) and bi-directed edges (\leftrightarrow). Directed edges model causal ancestry relations, whereas bi-directed edges denote that neither variable causes the other. Each edge has two *marks* (or *orientations*), tails (-) and/or arrowheads (>). A wildcard mark (*) can be a tail or an arrowhead. An edge $X^* \rightarrow Y$ is *into* Y , whereas an edge $X \rightarrow Y$ is *out of* X .

A *path* p in a directed mixed graph G is a sequence of distinct vertices $p = \langle V_1, V_2, \dots, V_{|p|} \rangle$, s.t. for $1 \leq i < |p|$, V_i and V_{i+1} are adjacent in G . The first and last vertices on a path are called the *end-points* of the path. X is a *parent*

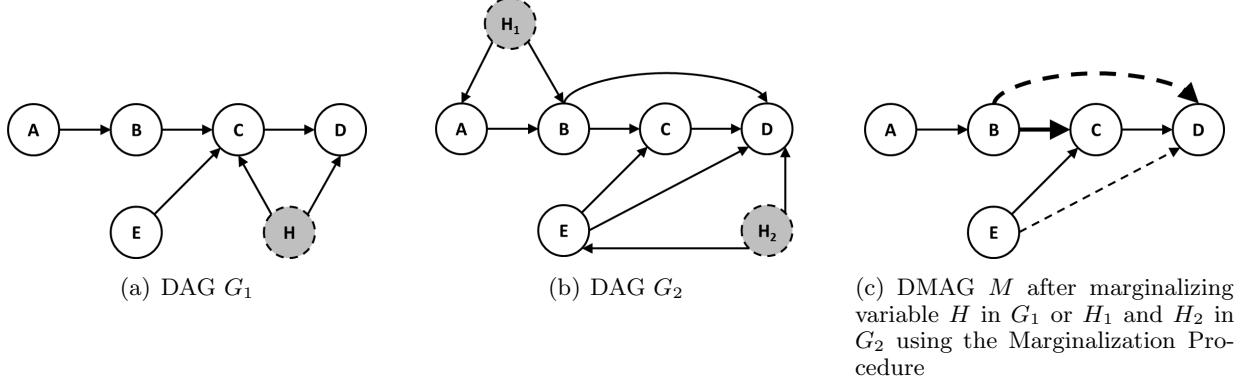


Figure 1: Directed edges in M are annotated: bold edges denote pure-causal relations; dashed edges denote possibly-indirect relations. For example, edge $A \rightarrow B$ in M is possibly-confounded (not bold) since there exist an extension DAG (G_2) where $A \leftarrow H \rightarrow B$. Edge $B \rightarrow D$ in M is possibly-indirect (dashed) since the edge is missing from G_1 . Edge $E \rightarrow D$ in M is both possibly-indirect and possibly-confounded: it is missing from G_1 and it is confounded in G_2 . Edge $B \rightarrow C$ is both a pure-causal and a direct-causal edge: there is no extension DAG such that B and C are confounded or the observed variables mediate the causation.

of Y and Y a *child* of X if $X \rightarrow Y$ is in G . A path p is *directed* if for $1 \leq i < |p|$, V_i is a parent of V_{i+1} in G . X is an *ancestor* of Y and Y a *descendant* of X if there is a directed path from X to Y in G . A *directed cycle* occurs in G if $Y \rightarrow X$ and X is an ancestor of Y . An *almost directed cycle* occurs in G if $Y \leftrightarrow X$ and X is an ancestor of Y . A triple $\langle X, Y, Z \rangle$ is said to form a *collider* if X and Z are into Y . A path p is called a *collider path* if every subsequent triple $\langle V_{i-1}, V_i, V_{i+1} \rangle$, $1 < i < |p|$ in p is a collider.

Definition 1 (Inducing Path). A path p is *inducing* relative to a set of vertices \mathbf{L} if: (a) every non-endpoint vertex on p is either in \mathbf{L} or a collider, and (b) every collider on p is an ancestor of an end-point vertex of the path. If the set of vertices \mathbf{L} is empty, the path is called a *primitive* inducing path.

Definition 2 (Directed Maximal Ancestral Graphs). A directed mixed graph is called a Directed Maximal Ancestral Graph (DMAG) if: (a) the graph does not contain any directed cycles, (b) the graph does not contain any almost directed cycles, and (c) for every pair of non-adjacent vertices in the graph, there is no primitive inducing path between them.

First, we want to point out that the definition

of inducing paths presented here differs from the standard definition of inducing paths (Richardson and Spirtes, 2002), since we don't deal with selection variables in this paper. The first two conditions imply that in a DMAG *an arrowhead denotes non-ancestry*. If X and Y are adjacent, and X is into Y , Y cannot be an ancestor of X because it would violate one of the conditions (if $X \rightarrow Y$, there would be a directed cycle, if on the other hand $X \leftrightarrow Y$, there would be an almost directed cycle). The third condition implies that two vertices are adjacent if and only if there is a primitive inducing path between them (Richardson and Spirtes, 2002). Notice that all conditions are met by Directed Acyclic Graphs (DAGs). Thus, DAGs are DMAGs without bi-directed edges. A DMAG M over variables $\mathbf{O} \setminus \mathbf{L}$ can be constructed by a DAG (or DMAG) G over variables \mathbf{O} and a set of latent variables \mathbf{L} as follows (Zhang, 2008):

Procedure 1 (Marginalization). (1) M contains the same variables as G except the latent variables \mathbf{L} , (2) two variables X and Y are adjacent in M if and only if there is an inducing path between them relative to \mathbf{L} in G , and (3) for two adjacent variables X and Y in M , the edge is oriented as (i) $X \rightarrow Y$ if X is an ancestor of Y in G , (ii) $X \leftarrow Y$ if Y is an ancestor of

X in G and, (iii) $X \leftrightarrow Y$ otherwise.

Figure 1(c) shows the marginal of the DAGs in Figure 1(a) or 1(b), where $L = \{H\}$ and $L = \{H_1, H_2\}$ respectively. It can be shown that a DMAG M created by this procedure represents the independencies of the marginal distribution entailed by the given G (Richardson and Spirtes, 2002) and additionally represents the causal semantics of G (since it retains the ancestral relations). We call M the **marginal graph** of G over \mathbf{L} and G an **extension** of M .

3 Problem Definition

Different DAGs may be represented by the same marginal DMAG over some observed variables for different sets of latent variables \mathbf{L} . Examples are shown in Figure 1. While the Marginalization Procedure ensures that the resulting DMAG represents the same set of conditional independencies among the observed variables, it does not distinguish between different *causal interpretations* of the edges. To facilitate the causal interpretation we define and subsequently solve the following problems:

Problem 1. Given a DMAG M , classify all directed edges $X \rightarrow Y$ into the following categories: **Pure-causal** if there is no extension DAG G of M such that $X \leftarrow H \rightarrow Y$, for some latent variable H , and **Possibly-confounded** if there is an extension DAG G where $X \leftarrow H \rightarrow Y$ for some latent variable H .

We note that (see Section 4 for more details), when an edge is possibly-confounded there is no way of determining whether it is actually confounded or not, solely by examining the graph (there may be other ways to induce the presence of possible confounders when assuming specific functional relations and noise distributions, e.g., (Peters et al., 2012)).

Another problem is that sometimes an edge may be present in a DMAG, while not being present in an extension DAG. Thus, this edge does not represent a direct causal relation in the context of the observed variables. This could happen because of inducing paths: two vertices are adjacent if there is an inducing path between them. We now define the following problem:

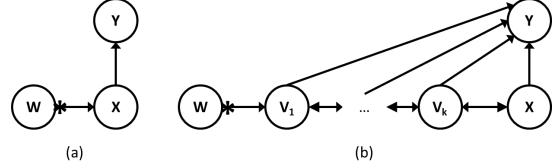


Figure 2: All cases of pure-causal edges. The “*” denotes that the end-point can be either tail or arrowhead. If any of the above induced subgraphs is present in a DMAG, edge $X \rightarrow Y$ is pure-causal.

Problem 2. Given a DMAG M , classify all directed edges $X \rightarrow Y$ into the following categories: **Direct-causal** if the edge is present in all extension DAGs of M , and **Possibly-indirect** if there is an extension DAG G of M that does not include the edge.

Direct-causal edges $X \rightarrow Y$ are important because they imply that the causal effect of X is present even when all other observed variables are held fixed (for some appropriate joint combination of their values). When an edge is possibly-indirect it is impossible to determine, solely by analyzing the graph, whether it is actually indirect or not.

4 Labeling Pure-causal and Possibly-Confounded Edges

The theory to distinguish between pure-causal and possibly-confounded directed edges is given in (Zhang, 2008). Actually, in (Zhang, 2008) the concepts of *visible* and *invisible* edges are defined, and subsequently, it is proved that they correspond to our definitions of pure-causal and possibly-confounded edges. The definition now follows:

Definition 3 (Visibility). A directed edge $X \rightarrow Y$ is *visible* if there is a vertex W not adjacent to Y , such that either there is an edge between W and X that is into X , or there is a collider path between W and X that is into X and every vertex on that path is a parent of Y . Otherwise $X \rightarrow Y$ is said to be *invisible*.

All possible cases of visible (pure-causal) edges are shown in Figure 2. (Zhang, 2008)

Algorithm 1 Find pure-causal Edges(DMAG \mathcal{M})

```

1: Mark all directed edges as possibly-confounded
2:  $Q \leftarrow \text{Queue}()$ 
3: for each  $X \rightarrow Y \in \mathcal{M}$  do
4:   if  $\exists W$  s.t.  $W* \rightarrow X \wedge \neg\text{adjacent}(W, Y)$  then
5:     Mark  $X \rightarrow Y$  as pure-causal
6:      $Q.\text{enqueue}(\{X, Y\})$ 
7:   end if
8: end for
9: while  $\neg Q.\text{isEmpty}()$  do
10:    $\{X, Y\} = Q.\text{dequeue}()$ 
11:   for each  $V$  s.t.  $X \leftrightarrow V \wedge V \rightarrow Y$  do
12:     if  $V \rightarrow Y$  is possibly-confounded then
13:       Mark  $V \rightarrow Y$  as pure-causal
14:        $Q.\text{enqueue}(\{V, Y\})$ 
15:     end if
16:   end for
17: end while
18: return All Marked Edges

```

proves that if a directed edge $X \rightarrow Y$ is visible then there is no inducing path between X and Y relative to the set of hidden variables that is into X . Consequently, X and Y cannot share a hidden common cause. Thus visibility is sufficient for $X \rightarrow Y$ to be pure-causal. Note that this does not imply that there are no other inducing paths between X and Y and so a visible edge can still be indirect: there can still be inducing paths relative to other sets of variables. Another important fact established by (Zhang, 2008) is that if $X \rightarrow Y$ is invisible, there is at least one DAG extension of the given DMAG, in which X and Y share a hidden common cause. As a result, visibility is also necessary for X and Y to be pure-causal. Thus, the two concepts are equivalent and no further distinction is necessary.

Algorithm 1 finds all pure-causal edges in a DMAG. We will prove that it is sound and complete and provide an upper bound for its time complexity.

Theorem 1. *Algorithm 1 is **sound** and **complete**.*

Proof. We will call a pure-causal path of length j for the edge $V_j \rightarrow Y$ a path of the form $W* \rightarrow V_1 \leftrightarrow \dots \leftrightarrow V_j \rightarrow Y$ and $V_i \rightarrow Y$, for all $i = 1, \dots, j$. As discussed, an edge $V_j \rightarrow Y$ is pure-causal if and only if there exists a pure-causal path of any length. We will use induction on the maximum length k of a causal path. For $k = 1$, the first For loop of the algorithm identifies all pure-causal edges by enumeration; these are the cases in Figures 2(a) and 2(b). Let us assume (inductive hypothesis) that the algorithm marks as pure-causal all edges $V_j \rightarrow Y$ that have a pure-causal path of length at most k . In the While loop, an edge $V_{j+1} \rightarrow Y$ will be marked as pure-causal if and only if (a) the subgraph $V_j \leftrightarrow V_{j+1} \rightarrow Y$ is present, and (b) $V_j \rightarrow Y$ is in the queue and thus has been marked pure-causal. By the inductive hypothesis and (b) above, there exist a pure-causal path of length at most k . By (a) this path can be extended to a sequence $W* \rightarrow V_1 \leftrightarrow \dots \leftrightarrow V_j \leftrightarrow V_{j+1}$, s.t., $V_i \rightarrow Y$, for all $i = 1, \dots, j+1$. If the sequence is not a path, then V_{j+1} coincides with some other node V_i , $1 \leq i \leq j$, and thus it has a pure-causal path of length strictly less than $k+1$. In that case, it would have already been marked and would not have passed the test at Line 12. In any case, there exist a causal-path of length at most $k+1$ for the marked edge $V_{j+1} \rightarrow Y$. \square

Theorem 2. *Algorithm 1 has $O(n \cdot m)$ time complexity.*

Proof. The computational costly parts of the algorithm are at lines 3-8 and 9-17. Note that each operation on the queue takes constant time $O(1)$.

The loop at line 3 will run $O(m)$ times. For the if statement at line 4, we have to find all vertices W , s.t. $W \rightarrow X$ or $W \leftrightarrow X$ is in the given DMAG, which can be computed in $O(n)$ time (each vertex is adjacent to at most $n - 1$ other vertices), and then out of those check if any is not adjacent to Y , which takes $O(1)$ time for fixed vertex W . The statements at lines 5 and 6 take constant time. Thus, lines 3-8 take $O(n \cdot m)$ time.

At this point, Q may contain at most $O(m)$ elements. At each iteration we remove one ele-

ment (line 10). Line 11 takes $O(n)$ time (similar to line 4). Lines 12 and 13 take constant time. The only line that could cause a problem is line 14, because it increases the number of elements in Q and could lead to additional iterations. Notice however that an element is added to the queue only if a pure-causal edge is found which was previously marked as possibly-confounded (line 12). But if it was possibly-confounded, it never was in Q . Therefore Q will contain each pure-causal edge **exactly once**. But the number of pure-causal edges is at most $O(m)$. Thus, the outer loop (line 9) will run $O(m)$ times, with a cost of $O(n)$ each, and the total running time of lines 9-17 is $O(n \cdot m)$.

Both, lines 3-8 and 9-17 take $O(n \cdot m)$ time, therefore the algorithm *Find pure-causal Edges* runs in $O(n \cdot m)$ time. \square

5 Definitely Direct and possibly-indirect Edges

An edge $X \rightarrow Y$ may not correspond to a direct causal relation, e.g., the edge $B \rightarrow D$ in the DMAG shown in Figure 1(c) does not exist in a DAG extension of that DMAG shown in Figure 1(a). Given a DAG G that misses the edge $X \rightarrow Y$, a set of latent variables L and the marginal DMAG M relative to L , an indirect edge $X \rightarrow Y$ appears in M when (a) there is a directed path $X \rightarrow \dots \rightarrow Y$ in G and (b) there exists an inducing path relative to L between X and Y in G . This follows directly from the Marginalization Procedure.

We now address the question, how such edges can be identified. First, notice that given a DMAG M with an indirect edge $X \rightarrow Y$ there always exists a DAG extension G which contains the edge $X \rightarrow Y$ (simply include the edge $X \rightarrow Y$ in G). Thus, it is impossible to identify whether a directed edge is indirect by observing only the DMAG. It is only possible to tell whether an edge is possibly-indirect (i.e. if there is a DAG extension in which the edge does not exist) or definitely direct (i.e. if it exists in all DAG extensions). It turns out that for given DMAG M , a directed edge $X \rightarrow Y$ is possibly-indirect if (a) there is a directed path from X

to Y in M and (b) an inducing path is possible between X and Y , i.e. there is an extension of M where it could happen. Notice that the edge $X \rightarrow Y$ in M is a directed path from X to Y and an inducing path between X and Y . Thus we have to check whether there are *non-direct* directed and inducing paths (i.e. without considering the edge $X \rightarrow Y$).

It is easy to check whether the first condition holds; since the Marginalization Procedure preserves ancestral relations, one has to check whether there is a non-direct directed path from X to Y . On the other hand, it is not obvious how the second condition can be checked when only observing a DMAG. The reason is that not all latent variables are encoded as bi-directed edges in the DMAG. As we saw in the last section however, we can distinguish between pure-causal and possibly-confounded edges, which comes handy here.

We split the problem solution to two separate cases. Let's first consider the case where a directed edge $X \rightarrow Y$ in a DMAG M is labeled as possibly-confounded. In this case, a non-direct inducing path trivially exists between X and Y in a DAG extension G of M ; just include a latent confounder H of X and Y in G , which creates the inducing path $X \leftarrow H \rightarrow Y$ in G relative to $L = \{H\}$. Thus, the edge is possibly-indirect if there also exists a non-direct directed path from X to Y in M , and is definitely direct otherwise. Now we turn our attention to the other case, i.e. to directed edges that are labeled as pure-causal.

Theorem 3. *Let $X \rightarrow Y$ be a pure-causal edge in a DMAG M . Then $X \rightarrow Y$ is possibly-indirect if and only if there is a vertex W in M s.t. $X \rightarrow W \rightarrow Y$ is in M and $W \rightarrow Y$ is possibly-confounded.*

Proof. If. If $X \rightarrow W \rightarrow Y$ is in M and $W \rightarrow Y$ is possibly-confounded, then there is a DAG extension of M where W and Y share a latent confounder $L_{W,Y}$, and thus the path $\langle X, W, L_{W,Y}, Y \rangle$ is inducing relative to $\mathbf{L} = \{L_{W,Y}\}$. Also the path $\langle X, W, Y \rangle$ is directed from X to Y . Thus *if $X \rightarrow W \rightarrow Y$ is in M then $X \rightarrow Y$ is possibly-indirect*.

And only if. For the opposite direction, assume that $X \rightarrow Y$ is possibly-indirect. Then by definition there exists a DAG extension G of M containing a non-direct directed path from X to Y and a non-direct inducing path p_{ind} from X to Y relative to some set of latent variables \mathbf{L} .

By definition of inducing paths, every non-latent vertex on p_{ind} is a collider on that path and an ancestor of X or Y . Since X is an ancestor of Y ($X \rightarrow Y$ is in M), *every collider on p_{ind} is an ancestor of Y* (if it is an ancestor of X it also is an ancestor of Y). Thus, it is easy to see that there is an inducing path from every non-latent vertex on p_{ind} to Y . As a result $X \rightarrow W \rightarrow Y$ or $X \leftarrow *W \rightarrow Y$ is in M (where W is the vertex which is adjacent to X and on p_{ind}). However, $X \leftarrow *W \rightarrow Y$ cannot be in M because it implies that X and W share a latent confounder in G (otherwise p_{ind} would not be an inducing path), which contradicts the fact that there is no inducing path from X to Y relative to the set of latent variables that is into X , because $X \rightarrow Y$ is pure-causal (given) (see Lemma 9 in (Zhang, 2008))

Thus, if $X \rightarrow Y$ is possibly-indirect, then $X \rightarrow W \rightarrow Y$ is in M . It remains to show that $W \rightarrow Y$ is always possibly-confounded. We will show this by contradiction.

Assume that $W \rightarrow Y$ is pure-causal. Then there exists a vertex U in M , s.t. U is not connected to Y and (a) $U \rightarrow W$ is in M , or (b) there is a collider path p_{col} in M from U to W where each vertex is a parent of Y .

Case (a): By replacing X with U in p_{ind} we create an inducing path from U to Y (each non-latent vertex is a collider and an ancestor of Y), thus U would be adjacent to Y in M and $U \rightarrow W$ would not be pure-causal, contradicting our assumption.

Case (b): Let p'_{col} be the path resulting by adding a vertex $L_{V_i, V_{i+1}}$ between each subsequent pair of vertices V_i and V_{i+1} in p_{col} , where $L_{V_i, V_{i+1}}$ is a latent parent of V_i and V_{i+1} which replaces the bi-directed edge $V_i \leftrightarrow V_{i+1}$ in G . Then there is an inducing path $p'_{ind} = \langle p'_{col}, p_{ind} \setminus X \rangle$, since (a) every non-latent vertex on p'_{col} is a collider, (b) every non-latent vertex on p_{ind} is a collider, (c) W is a collider (the last

vertex on p'_{col} and the first vertex on $p_{ind} \setminus X$ are into W), (d) every vertex on p'_{col} is a parent of Y , and (d) every vertex on $p_{ind} \setminus X$ is an ancestor of Y , thus U and Y would be adjacent in M and $U \rightarrow W$ would not be pure-causal, contradicting our assumption.

Thus, if $X \rightarrow Y$ is possibly-indirect, then $X \rightarrow W \rightarrow Y$ is in M and $W \rightarrow Y$ is possibly-confounded. \square

Theorem 4. *The time complexity of labeling all directed edges in a MAG \mathcal{M} as direct-causal or possibly-indirect, is $O(n \cdot m + n^2)$.*

Proof. As a first step, we have to label each directed edge as pure-causal or possibly-confounded. This takes $O(n \cdot m)$ time (Theorem 2). In order to avoid unnecessary computations we pre-compute the ancestral relations (transitive closure) of \mathcal{M} , by ignoring bi-directed edges. This can be trivially done in $O(n \cdot m + n^2)$ time. To label all possibly-confounded edges, we have to consider each edge once, and since all ancestral relations are pre-computed the total time is $O(m)$. Labeling all pure-causal edges takes $O(n \cdot m)$ time, since for each edge $X \rightarrow Y$ we have to consider each node at most once (to check if it is connected to X and Y), and we have to check whether the remaining conditions are satisfied (see Theorem 3), which can be done in constant time. The total time complexity is $O(n \cdot m + n^2)$. \square

6 Quantifying the Frequency of the Pure-Causal and Direct-Causal Edges

Experimental Setup. We fixed the number of vertices to $n = 40$ and repeated the following experiment 500 times: (a) We varied the edge density d from 0.025 to 1, with a step size of 0.025. For k nodes, the number of edges m is $m = \text{round}(d \cdot k \cdot \frac{k-1}{2})$. (b) We varied the number of latent variables l (not necessarily latent confounders) from 0 to 20. (c) We generated uniformly at random a DAG G with $n + l$ nodes and m edges (defined above). The set of latent variables \mathbf{L} was generated by randomly picking l vertices. A

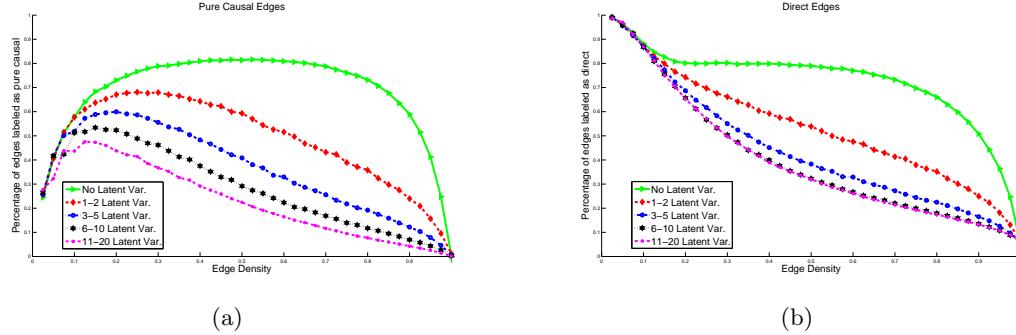


Figure 3: (a) Frequencies of pure-causal and direct-causal edges in randomly generated DMAGs. A relatively high percentage of directed edges can be labeled as pure-causal for graphs with intermediate density. (b) Similarly, a high percentage of directed edges can be labeled as direct-causal for low and medium graph densities.

DMAG M was generated given G and \mathbf{L} using the Marginalization Procedure. Thus, in each DMAG we observe 40 variables, whereas the number of latent variables varies between 0 and 20. (d) We identified all pure causal and direct edges for M . The total number of runs is $500 \cdot 40(\text{density}) \cdot 21(\text{latent variables}) = 420000$.

Results. The results are shown in Figure 3(a) and Figure 3(b). The x-axes are the edge density (both directed and bi-directed edges) of the resulting DMAG after marginalization of the generating DAG. The y-axes are the percentage of *directed edges* in the DMAG that are labeled as pure-causal or direct-causal respectively. The lines in the plots have been produced by a moving average of a window with range 0.01 and a step of 0.025 to smooth out the results.

There are several observations to make. A relatively high percentage of directed edges can be labeled as pure-causal for graphs with intermediate density (Figure 3(a)). The plots follow a bell shape with the percentage of pure-causal edges dropping to 0 for complete graphs. The percentage of pure-causal edges decreases as the number of latent variables increases. Similarly, a high percentage of directed edges can be labeled as direct-causal for low and medium graph densities (Figure 3(b)). The percentage of direct-causal edges decreases monotonically with increased graph density.

7 Conclusion

The causal interpretation of the directed edges $X \rightarrow Y$ in a Maximal Ancestral Graph is not straight-forward and could be misleading to an unfamiliar practitioner. We present efficient algorithms for the theory present by (Zhang, 2008) and extensions that can label directed edges to those that cannot be confounded (pure-causal) or those that cannot be mediated by other observed variables (direct-causal). The algorithms could facilitate interpretation and understanding of a causal graph when latent confounding variables are admitted. The percentage of pure-causal and direct-causal edges depends on the density of the causal graph.

References

- Z. Cai, M. Kuroki, J. Pearl, and J. Tian. 2008. Bounds on Direct Effects in the Presence of Confounded Intermediate Variables. *Biometrics*, 64:695–701.
- J. Pearl. 2000. *Causality, Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, U.K.
- J. Peters, J. M. Mooij, D. Janzing, and B. Schölkopf. 2012. Identifiability of causal graphs using functional models. *CoRR*, abs/1202.3757.
- Th. Richardson and P. Spirtes. 2002. Ancestral graph Markov models. *Annals of Statistics*, 30(4):962–1030.
- P. Spirtes, C. Glymour, and R. Scheines. 2000. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, 2nd edition.
- J. Zhang. 2008. Causal reasoning with ancestral graphs. *J. Mach. Learn. Res.*, 9:1437–1474.

Approximate Inference in Influence Diagrams using Binary Trees

Rafael Cabañas de Paz, Manuel Gómez-Olmedo, Andrés Cano

Dept. Computer Science and Artificial Intelligence

University of Granada, CITIC-UGR, Spain

rcabanas@decsai.ugr.es, mgomez@decsai.ugr.es, acu@decsai.ugr.es

Abstract

This paper introduces binary trees, a new kind of representation of the potentials involved in Influence Diagrams. This kind of tree allows representing context-specific independencies that are finer-grained compared to those encoded using other representations, such as numerical trees or tables. This enhanced capability can be used to improve the efficiency of the algorithms used for Influence Diagrams.

1 Introduction

Decision problems under uncertainty have traditionally been represented and solved using *Decision Trees* (Raiffa, 1968). However, they have a problem of exponential growth of the representation. An alternative are *Influence Diagrams* (IDs), which can encode the independence relations between variables in a way that avoids this exponential growth.

For complex decision problems, the evaluation of an ID becomes unfeasible due to its computational cost: The set of information states exceeds the storage capacity of PCs or the optimal policy must be obtained in a short period of time. It is thus necessary to use approximate methods for ID evaluation such as *LMIDs* (Lauritzen and Nilsson, 2001), or sampling techniques (Charnes and Shenoy, 2004; Cano et al., 2006). Some of the deterministic methods use alternative representations for potentials, such as *numerical trees* (NTs) (Cano et al., 2000). This representation offers the possibility of taking advantage of *context-specific independencies*. NTs can be pruned and converted into smaller trees when potentials are too large, thus leading to approximate algorithms. Here, we introduce a new kind of tree for the representation of potentials, namely, *binary trees* (BTs), where the internal nodes always have two children. These trees allow the specification of finer-grained context-specific independencies

than NTs, and should lead to more efficient algorithms.

The paper is organized in the following way: Section 2 introduces some concepts and notation about IDs; Section 3 presents basic concepts of NTs; Section 4 describes key issues about BTs and how they are used during the evaluation of IDs; Section 6 includes the experimental work and results; finally Section 7 details our conclusions and lines for future work.

2 Influence Diagrams

An ID (Olmsted, 1984) is a Bayesian network (BN) augmented with two new types of nodes: *decision nodes* (mutually exclusive actions which the decision maker can control) and utility nodes (representing decision maker preferences). Utility variables may depend on both random (or chance) variables and decision variables. IDs are used for representing and solving decision problems.

The set of chance nodes is denoted by V_C , the set of decision nodes is denoted by V_D , and the set of utility nodes is denoted by V_U . Direct predecessors of a decision node D are called *informational parents*. The set of all possible combinations of states of the informational parents is called the *information set* for D . The elements of this set are denoted *information states for D* . The *universe* of the ID

is $V = V_C \cup V_D = \{X_1, \dots, X_n\}$. Let us suppose that each variable X_i takes values on a finite set $\Omega_{X_i} = \{x_1, \dots, x_{|\Omega_{X_i}|}\}$. If I is a set of indexes, we shall write \mathbf{X}_I for the set of variables $\{X_i | i \in I\}$, defined on $\Omega_{\mathbf{X}_I} = \times_{i \in I} \Omega_{X_i}$. The elements of $\Omega_{\mathbf{X}_I}$ are called configurations of \mathbf{X}_I and will be represented as \mathbf{x}_I . A *probability potential* denoted by ϕ is a mapping $\phi : \Omega_{\mathbf{X}_I} \rightarrow [0, 1]$. An *utility potential* denoted by ψ is a mapping $\psi : \Omega_{\mathbf{X}_I} \rightarrow \mathbb{R}$.

The relevant past π_{D_i} for a decision variable D_i is the subset of the informational parents of D_i for making decision D_i . Then, a decision rule for D_i is a mapping $d_i : \Omega_{\pi_{D_i}} \rightarrow \Omega_{D_i}$. A strategy is an ordered set of decision rules $S = \{d_1, \dots, d_n\}$, including a decision rule for each decision. An optimal strategy \hat{S} returns the optimal choice the decision maker should take for each decision. To evaluate an ID we must compute an optimal strategy \hat{S} that maximizes the expected utility for the decision maker, and compute its maximum expected utility $MEU(\hat{S})$.

3 Numerical Trees

NTs have been used previously to represent potentials in BNs (Cano et al., 2000) and IDs (Gómez and Cano, 2003). Their main advantage is that they allow the specification of *context-specific independencies* (Boutilier et al., 1996). Moreover, a NT can be pruned in order to reduce its storage size (and this also decreases the execution time). Doing so will allow approximating the potentials. In general, NTs can be used to encode probability and utility functions, which will be called *numerical probability trees* (NPTs) and *numerical utility trees* (NUTs) respectively.

A NT defined over the set of variables \mathbf{X}_I is a directed tree, where each internal node is labelled with a variable (random variable or decision node), and each leaf node is labelled with a number (a probability or a utility value). We use L_t to denote the *label of node t*. Each internal node has an outgoing arc for each state of the variable associated with that node. Out-

going arcs from a node X_i are labelled with the same name of the state ($x_i \in \Omega_{X_i}$) associated to X_i . The *size* of a tree \mathcal{NT} , denoted $size(\mathcal{NT})$, is defined as its number of leaves. A subtree of \mathcal{NT} is a terminal tree if it contains one node labelled with a variable and all its children are leaf nodes.

4 Binary Trees

A BT is similar to a NT. It is also a directed labelled tree, where each internal node is labelled with a variable, and each leaf is labelled with a non-negative real number. It also allows representing a potential for a set of variables \mathbf{X}_I . But in this case each internal node has always two outgoing arcs, and a variable can appear more than once labelling the nodes in the path from the root to a leaf node. Another difference is that, for an internal node labelled with X_i , the outgoing arcs can usually be labelled with more than one state of Ω_{X_i} . We denote by $L_{lb(t)}$ and $L_{rb(t)}$ the labels (two subsets of $\Omega_{X_i}^t$) of the left and right branches of node t . Then, we denote by t_l and t_r the children of t . Trees for encoding probability functions will be called *binary probability trees* (BPTs) and trees for utility functions will be called *binary utility trees* (BUTs).

An advantage of BTs is that they allow representing context-specific independencies (*contextual weak independencies*) (Wong and Butz, 1999) which are finer-grained than those represented using NTs. For example, Figure 1 shows three different representations for an utility potential: Table (a), NUT (b), and BUT (c). When $A = a_1$, the potential will always take the value 30, regardless of the value of B . Therefore, less space is needed for representing it as a tree, since it can be pruned. Besides, an additional pruning can be made for the BUT: When $A = a_2$ and $B \in \{b_1, b_2\}$, the potential will always be 45.

4.1 Building a BUT

Building a BUT is an optimization problem that consists in choosing the labels for each internal

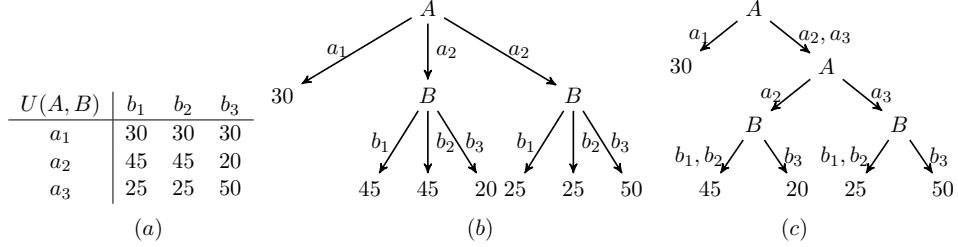


Figure 1: Utility potential represented as a table (a), as a NUT (b) and as BUT (c)

node (variable) and each arc (state). The order of the labels will affect the context-specific independencies represented by the tree. A previous work, (Cano et al., 2011), proposed a greedy algorithm to build a BPT from a given probability potential. The algorithm for building a BUT, which is quite similar, is described in this subsection.

The algorithm builds the BUT using a top-down approach, choosing at each step a variable and two partitions of its states. The process begins with an initial \mathcal{BUT}_0 which has only one node labelled with the average of the values in the potential: $L_t = \sum_{\mathbf{x}_I \in \Omega_{\mathbf{X}_I}} \psi(\mathbf{x}_I) / |\Omega_{\mathbf{X}_I}|$.

A greedy step is then applied successively until an exact BUT is obtained. At each step, a new \mathcal{BUT}_{j+1} is generated from the previous one, \mathcal{BUT}_j . This new tree is the result of expanding one of the leaf nodes t in \mathcal{BUT}_j with a terminal tree (where t roots the terminal tree, and t_l and t_r are children of t). Node t is labelled with one of the *candidate variables*. The set of available states $\Omega_{X_i}^t$ of the chosen candidate variable X_i are partitioned into two subsets, $\Omega_{X_i}^{t_l}$ and $\Omega_{X_i}^{t_r}$. Each subset labels one of the two outgoing arcs (left and right) of t . The two leaf nodes t_l and t_r in the new terminal tree are labelled with the average of ψ values consistent with the states labelling the path from the root to the terminal node.

The trees built at every step can be considered approximations of the potential. Therefore, it is required a distance to measure the goodness of the approximation of a given potential ψ represented by a \mathcal{BUT} . The Euclidean

distance is proposed for that purpose:

$$D(\psi, \mathcal{BUT}) = \sqrt{\sum_{\mathbf{x}_i \in \Omega_{\mathbf{X}_I}} (\psi(\mathbf{x}_i) - \mathcal{BUT}(\mathbf{x}_i))^2} \quad (1)$$

At each step, a variable and two state partitions must be chosen in order to maximize the *information gain*. Thus, variables in a BUT may not appear in the same order in every path from root to leaf. The information gain can be defined as the difference between the distances of two approximations.

Definition 1 (Information Gain). Let ψ be the potential we are constructing and $\mathcal{BUT}_j(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r})$ the tree resulting of expanding the leaf node t with the candidate variable X_i and a partition of its available states into sets $\Omega_{X_i}^{t_l}$ and $\Omega_{X_i}^{t_r}$. The *information gain* can be defined as:

$$\begin{aligned} I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) &= \\ &= D(\psi, \mathcal{BUT}_j) - D(\psi, \mathcal{BUT}_j(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r})) \end{aligned} \quad (2)$$

In our experiments (Section 6) we did not check every possible partition of $\Omega_{X_i}^t$, because this would be a very time-consuming task. Assuming that the set of available states for X_i at node t is ordered, we will only check partitions into subsets with consecutive states. Thus, each expansion is defined by a variable and a splitting point.

4.2 Pruning a BUT

If the size of a BUT needs to be reduced, it can be pruned in order to get a new BUT which approximates the potential. Pruning a BUT con-

sists in replacing a terminal tree by the average value of its leaves.

Definition 2 (Pruning a terminal tree). Let \mathcal{BUT} be a binary utility tree encoding ψ , t the root of a terminal tree labelled with X_i , t_c and t_r its children, $\Omega_{X_i}^{t_l}$ and $\Omega_{X_i}^{t_r}$ the sets of states for left and right child respectively, $\max(\psi)$ and $\min(\psi)$ the maximum and minimum values in ψ , and Δ a given threshold $\Delta \geq 0$. Then the terminal tree rooted by X_i can be pruned if:

$$I(t, X_i, \Omega_{X_i}^{t_l}, \Omega_{X_i}^{t_r}) \leq \Delta \cdot (\max(\psi) - \min(\psi)) \quad (3)$$

The goal of pruning involves detecting leaves that can be replaced by one value without a great increment in the distance between the approximation and the exact tree.

4.3 Operations with BUTs

Inference algorithms for IDs require five operations with potentials: Those used for BNs (*restriction*, *combination*, and *sum-marginalization*) were described in a previous work about their direct use with BTs (Cano et al., 2011). The evaluation of IDs also requires *max-marginalization* and *division*. Here we will detail the former and the auxiliary operation *max*. Division is similar to combination, so it is not described due to space restrictions.

When solving IDs, chance nodes are removed using sum-marginalization (as in BNs) while decision nodes are removed through max-marginalization. As it can be seen in Algorithm 1, the removal of X_j through max-marginalization produces a new tree denoted $\max_{X_j} \mathcal{BUT}$. This operation is similar to sum-marginalization, but instead of adding the values for X_j and a given configuration for $\mathbf{X}_i \setminus X_j$, it returns the maximum value. Figure 2 shows the application of these operations to a tree in order to remove variable B . The algorithm is recursively executed until a node labelled with the variable to be removed is found. When it happens, the algorithm max-marginalizes the left and right children trees and combines them using the *max* operation, described in Algorithm 2. The *max* operation returns a potential containing the maximum for each configuration of

```

input :  $t$  (root node of  $\mathcal{BUT}$ );
       $X_j$  (variable to be removed)
output: the root of  $\max_{X_j} \mathcal{BUT}$ 
if  $t$  is a leaf node then
    Build a new node  $tn$ ;
    Set  $L_{tn} = L_t$  the label of  $tn$ ;
end
else
    if  $L_t == X_j$  then
         $t1 = \text{max-marginalize}(t_l, X_j);$ 
         $t2 = \text{max-marginalize}(t_r, X_j);$ 
         $tn = \max(t1, t2);$ 
    end
    else
        Build a new node  $tn$ ;
        Set  $L_{tn} = L_t$ ;
        Set  $L_{lb(tn)} = L_{lb(t)}$ ;
        Set  $L_{lr(tn)} = L_{lr(t)}$ ;
        Set  $\text{max-marginalize}(t_l, X_j)$  the left
        child of  $tn$ ;
        Set  $\text{max-marginalize}(t_r, X_j)$  the
        right child of  $tn$ ;
    end
end
return  $tn$ ;

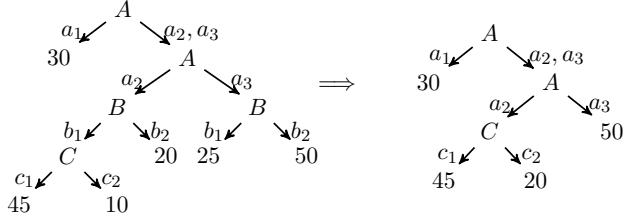
```

Algorithm 1: max-marginalization

the variables involved. This operation requires restricting a \mathcal{BUT} to a set of states L of a variable X_i , denoted $\mathcal{BUT}^{R(X_i, L)}$.

5 IDs Inference

Inference algorithms for IDs can be easily adapted for working with BTs. The adaptations only requires using BTs and their related operations for computing. Here we propose the Variable Elimination (VE) algorithm using BTs. This method can be used for solving BNs (Zhang and Poole, 1996) and IDs (Jensen and Nielsen, 2007). A pruning process can be performed to obtain smaller trees, thus reducing the computing time. This pruning is performed only with the initial utility potentials present in the ID. After that, the inference algorithm is the same.

Figure 2: max-marginalization of a BUT with respect to variable B

```

input :  $t_1$  and  $t_2$  (root nodes of  $\text{BUT}_1$  and  $\text{BUT}_2$ );
output: the root of  $\max(\text{BUT}_1, \text{BUT}_2)$ 
Build a new node  $t$ ;
if  $t_1$  is a leaf node then
  if  $t_2$  is a leaf node then
    if  $t_1 > t_2$  then
      |  $L_t = L_{t_1}$ 
    end
    else
      |  $L_t = L_{t_2}$ 
    end
  end
  else
    Set  $L_t = L_{t_2}$ ;
    Set  $L_{lb(t)} = L_{lb(t_2)}$ ;
    Set  $L_{rb(t)} = L_{rb(t_2)}$ ;
    Set  $\max(t_1, t_{2l})$  the left child of  $t$ ;
    Set  $\max(t_1, t_{2r})$  the right child of  $t$ ;
  end
end
else
   $X_i = L_{t_1}$ ;
  Set  $L_t = L_{t_1}$ ;
  Set  $L_{lb(t)} = L_{lb(t_1)}$ ;
  Set  $L_{rb(t)} = L_{rb(t_1)}$ ;
  Set  $\max(t_{1l}, \text{BUT}_2^{R(X_i, L_{lb(t_1)})})$  the left child of  $t$ ;
  Set  $\max(t_{1r}, \text{BUT}_2^{R(X_i, L_{rb(t_1)})})$  the right child of  $t$ ;
end
return  $t$ 
Algorithm 2:  $\max$ 

```

6 Experiments

In this section, the performance of NTs and BTs for IDs inference is analyzed. However, we

must introduce first some concepts about *Multi-objective Optimization Problems*.

6.1 Multi-Objective Optimization Problems

In the problem of approximating a potential tree, there are two objectives to be considered: Size and error of the approximated potential tree. These two objectives can be controlled using the Δ threshold for pruning: A low Δ value will produce large trees with a low error, while a high Δ value will produce small trees with a big error. Thus, it can be considered as a Multi-Objective Optimization Problem (MOP) (Jin and Sendhoff, 2008) with two objectives to be minimized. In this kind of problems, there is a set of objective functions that must be optimized. Hence, optimizing means finding such a solution with acceptable values for all the objectives.

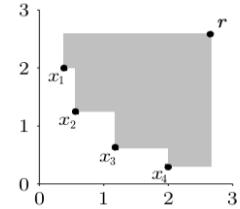


Figure 3: Hypervolume for a minimization problem

Without additional information about the preferred objective and level of performance, several possible solutions may exist. Moreover, the objectives considered are usually in conflict. In MOPs, the set of acceptable solutions composes the Pareto set (non-dominated solutions). In order to compare two solution sets, we used the Hyper-volume indicator, (Zitzler et

al., 2007). It is a unary indicator that measures the area of the dominated portion of the space given a reference point r (see Figure 3). It is defined in the interval $[0, 1]$, being 1 the optimal solution. For minimization problems (this is our case) the reference point is the maximum at every dimension.

6.2 Procedure

The aim of our experiments was to test the improvements offered by BUTs with respect to NUTs during ID evaluation. For that purpose, we used different kinds of IDs:

- A real world ID used for the treatment of gastric NHL disease (Bielza et al., 2008) with 3 decisions, 1 utility node and 17 chance nodes.
- A set of 30 randomly generated IDs with 2 decisions, 1 utility node, and a random number (between 6 and 17) of chance nodes. The utility function was created using a random biased number generator.

The inference algorithm employed for the test was VE. In order to check the error produced by pruning the trees, utility trees were initially transformed through this operation. Probability trees were not pruned, since the focus was on manipulating utility trees. The Δ threshold used for pruning was ranged in the interval $[0, 1]$ (see Equation 3).

The utility potentials analyzed to check the error were the initial ones present in the ID and the expected utility for each decision. These are tested by measuring their size and the *root-mean-square error* (RMSE) with respect to the exact values obtained using a Δ threshold of 0. Each ID was evaluated using NUTs and BUTs with different threshold values. For each evaluation, the size and the RMSE were measured. All the pairs (size, RMSE) for the same ID and kind of tree compose a solution set. For each solution set, the Pareto front and hyper-volume indicator were computed. Finally, a Wilcoxon signed-rank test was performed with the hyper-volume values from random IDs. The null hypothesis was that using NUTs or BTs produced

the same hyper-volume. The significance level for rejecting the null hypothesis was 5%.

6.3 Results

The experiments showed that BUTs offer better approximate solutions than NUTs. The same error level will be achieved using BUTs of smaller size than the corresponding NUTs. This situation can be shown in Figure 4, which includes four different graphics representing RMSE (horizontal axis) against tree size (vertical axis) for initial utilities and policies corresponding to decisions 0, 1 and 2 for the NHL ID. An additional advantage of BUTs is that the number of different solutions obtained is higher. This is due to the possibility of doing a softer pruning operation on them, which can only act on a certain subset of states gathered in a given branch.

Table 1 shows the hyper-volume indicators obtained from the evaluation of the NHL ID. Each row corresponds to one of the utility potentials (initial utility and expected utilities), whereas each column corresponds to the kind of tree (NUT or BUT) used to encode the potential. It can be shown that the hyper-volume value (H_B) for a binary tree is always higher than the corresponding hyper-volume value (H_N) for a numerical tree. Only the initial utilities return similar values. Moreover, for the last decision removed (Decision 0), the difference is more significant. Thus, better approximations are achieved using BUTs.

	H_N	H_B
Initial Utility	0.737	0.795
Decision 2	0.252	0.829
Decision 1	0.248	0.991
Decision 0	0	0.915

Table 1: Hyper-volume values of approximate utility trees comparing NTs and BTs for the NHL Influence Diagram

The hyper-volume values for the random IDs are shown in Table 2. Each row corresponds to a different random ID, whereas the columns indicate the kind of tree (NUT or BUT) and util-

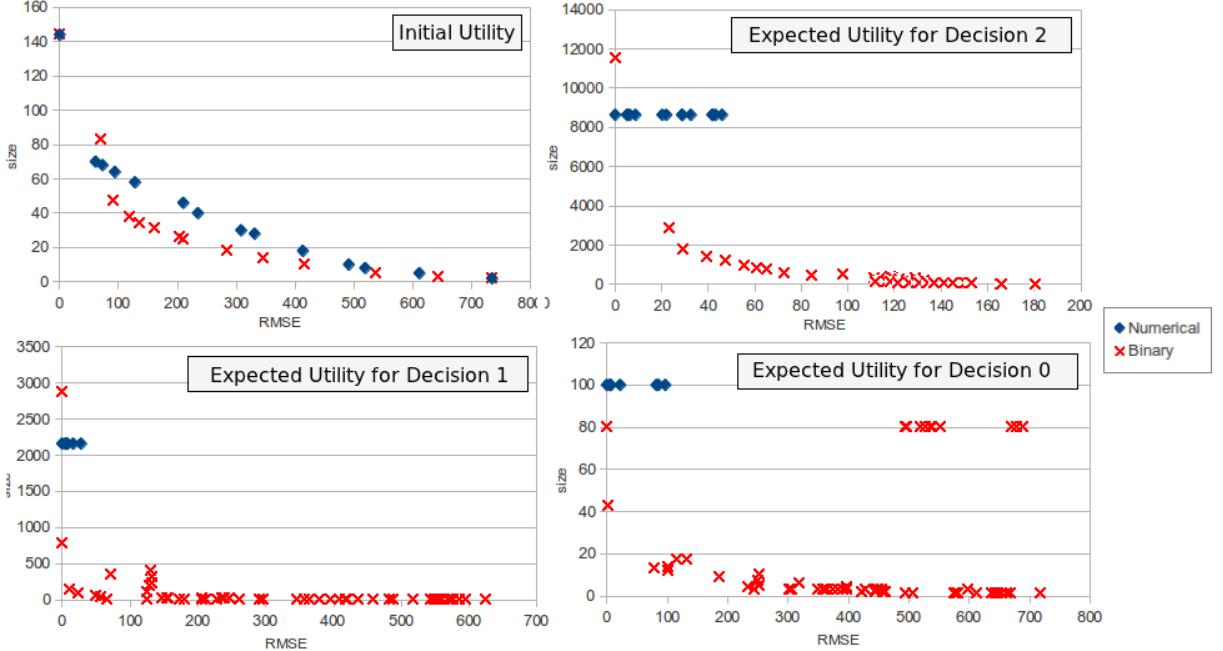


Figure 4: Results of utility trees approximation for the NHL Influence Diagram.

ity potential (initial utility and expected utilities) analyzed. It can be observed that the BT hyper-volume (H_B) is usually higher than the corresponding NT hyper-volume (H_N).

The results of the Wilcoxon signed-rank test comparing H_N and H_B obtained from the evaluation of random IDs are shown in Table 3. For each utility potential, it shows the *p-value* and whether the null hypothesis was rejected (NTs and BTs are not equal) with a significance level of 5%. The null hypothesis was rejected for all the utility potentials. That is, the results for BUTs are better than those for NUTs.

7 Conclusions and future work

In this paper, we have introduced a new type of utility potential representation: BTs. The experiments showed that BTs offer better approximate solutions than NTs. The same error level will be achieved using a BT of smaller size than the corresponding NT. As regards future directions of research, we can study the behaviour of BTs using alternatives to the VE inference algorithm, like *Arc Reversal* (Shachter, 1986), *Lazy propagation in clique trees* (Madsen and Jensen, 1999), etc. Another interesting

possibility would be to analyze the size of intermediate potentials and the effects of pruning utility trees. Finally, another direction of research could be the integration of restrictions with BTs. This would allow the treatment of asymmetric decision problems.

Acknowledgments

This research was supported by the Spanish Ministry of Economy and Competitiveness under projects TIN2010-20900-C04-01, the European Regional Development Fund (FEDER) and the FPI scholarship programme (BES-2011-050604). The authors have been also partially supported by “Consejería de Economía, Innovación y Ciencia de la Junta de Andalucía” under projects TIC-06016 and P08-TIC-03717.

References

- C. Bielza, J.A. Fernández del Pozo, and P.J.F. Lucas. 2008. Explaining clinical decisions by extracting regularity patterns. *Decision Support Systems*, 44(2):397–408.
- C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. 1996. Context-specific independence in Bayesian networks. In *Proceedings of the 12th International Conference on Uncertainty in AI*, pages 115–123. Morgan Kaufmann Publishers Inc.

Initial Utility		Decision 1		Decision 0	
H_N	H_B	H_N	H_B	H_N	H_B
0.635	0.811	0.84	0.892	0	0.907
0.744	0.888	0.646	0.675	0	0.85
0.813	0.872	0.795	0.781	0.556	0.781
0.604	0.767	0.792	0.783	0.84	0.733
0.513	0.742	0	0.806	0	0.805
0.67	0.861	0.588	0.93	0	0.807
0.656	0.841	0.781	0.808	0.92	0.941
0.567	0.811	0.75	0.826	0.764	0.826
0.797	0.877	0.799	0.965	0	0.686
0.51	0.68	0	0.893	0	0.746
0.654	0.879	0.307	0.885	0.405	0.912
0.464	0.461	0.613	0.558	0.61	0.526
0.588	0.75	0.572	0.816	0	0.588
0.6	0.799	0.686	0.848	0	0.835
0.453	0.535	0.685	0.714	0.373	0.453
0.693	0.77	0.775	0.861	0.6	0.608
0.217	0.422	0.406	0.581	0.376	0.126
0.636	0.895	0.829	0.917	0.783	0.905
0.464	0.671	0.611	0.813	0.614	0.71
0.282	0.361	0.283	0.36	0	0.148
0.772	0.926	0.866	0.868	0.892	0.961
0.215	0.422	0.406	0.577	0.377	0.125
0.659	0.865	0.887	0.914	0.515	0.95
0.485	0.668	0.416	0.771	0	0.537
0.491	0.611	0.635	0.78	0.698	0.697
0.219	0.562	0.381	0.687	0.387	0.0777
0.763	0.818	0.724	0.802	0.498	0.796
0.338	0.506	0.583	0.637	0.36	0.439
0.455	0.864	0.834	0.926	0.654	0.778
0.649	0.855	0.89	0.937	0.91	0.963

Table 2: Hyper-volume values of utility trees comparing NTs and BTs for each random ID

- A. Cano, S. Moral, and A. Salmerón. 2000. Penniless propagation in join trees. *International Journal of Intelligent Systems*, 15(11):1027–1059.
- A. Cano, M. Gómez, and S. Moral. 2006. A forward-backward Monte Carlo method for solving influence diagrams. *International Journal of Approximate Reasoning*, 42(1):119–135.
- A. Cano, M. Gómez-Olmedo, and S. Moral. 2011. Approximate inference in Bayesian networks using binary probability trees. *International Journal of Approximate Reasoning*, 52(1):49–62.
- J.M. Charnes and P.P. Shenoy. 2004. Multistage

	p-value	rejected
Initial Utility	$1.9 \cdot 10^{-6}$	yes
Decision 1	$8.5 \cdot 10^{-6}$	yes
Decision 0	0.001	yes

Table 3: Results of the Wilcoxon test against the results obtained using NTs and BTs

- Monte Carlo method for solving influence diagrams using local computation. *Management Science*, pages 405–418.
- M. Gómez and A. Cano. 2003. Applying numerical trees to evaluate asymmetric decision problems. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 196–207.
- F.V. Jensen and T.D. Nielsen. 2007. *Bayesian networks and decision graphs*. Springer Verlag.
- Y. Jin and B. Sendhoff. 2008. Pareto-based multi-objective machine learning: An overview and case studies. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):397–415.
- S.L. Lauritzen and D. Nilsson. 2001. Representing and solving decision problems with limited information. *Management Science*, pages 1235–1251.
- A.L. Madsen and F.V. Jensen. 1999. Lazy evaluation of symmetric Bayesian decision problems. In *Proceedings of the 15th Conference on Uncertainty in AI*, pages 382–390. Morgan Kaufmann Publishers Inc.
- S.M. Olmsted. 1984. Representing and solving decision problems. *Dissertation Abstracts International Part B: Science and Engineering*, 45(3).
- H. Raiffa. 1968. Decision analysis: introductory lectures on choices under uncertainty.
- R.D. Shachter. 1986. Evaluating influence diagrams. *Operations research*, pages 871–882.
- S.K.M. Wong and C.J. Butz. 1999. Contextual weak independence in Bayesian networks. In *Proceedings of the 15th conference on Uncertainty in AI*, pages 670–679. Morgan Kaufmann Publishers Inc.
- N.L. Zhang and D. Poole. 1996. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328.
- E. Zitzler, D. Brockhoff, and L. Thiele. 2007. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Evolutionary Multi-Criterion Optimization*, pages 862–876. Springer.

The Same-Decision Probability: A New Tool for Decision Making

Suming Chen and Arthur Choi and Adnan Darwiche

Computer Science Department

University of California, Los Angeles

{suming,aychoi,darwiche}@cs.ucla.edu

Abstract

When using graphical models for decision making, a fundamental question is whether one is ready to make a decision (stopping criteria), and if not, what observations should be made to better prepare for a decision (selection criteria). In this paper, we review the notions of entropy and expected utility, which are commonly used for this purpose, and contrast them with a newly introduced notion, called the *same-decision probability*, which can be used both as a stopping criteria for making decisions and as a selection criteria for choosing additional observations. Furthermore, we show that computing the same-decision probability lies in the same complexity class as a general expectation computation problem that is applicable to a wide variety of queries in graphical models, including the computation of non-myopic value of information.

1 Introduction

Probabilistic graphical models have often been used to model a variety of decision problems, e.g., in medical diagnosis (Pauker and Kassirer, 1980; van der Gaag and Coupé, 1999), fault diagnosis (Lu and Przytula, 2006), troubleshooting (Heckerman et al., 1995), and in intrusion detection (Kruegel et al., 2003). In these and similar applications, there are often unobserved variables, such as the state of a patient’s health, or the presence of a security breach, leading to two fundamental questions. The first question is whether, given the current observations, the decision maker is ready to commit to a decision. We will refer to this as the *stopping criteria* for making a decision. Assuming the stopping criteria is not met, the second question is what additional observations should be made before the decision maker is ready to make a decision. This typically requires a *selection criteria* based on some measure for quantifying an observation’s *value of information* (VOI).

The literature contains a number of proposals for both stopping and selection criteria. On stopping criteria, one may commit to a decision once the belief about a certain event crosses

some threshold, as in (Pauker and Kassirer, 1980; Lu and Przytula, 2006). Alternatively, we may simply perform as many observations as our budget allows, as in (Greiner et al., 1996; Krause and Guestrin, 2009). As for selection criteria, different observations may have different values with respect to the decision we are interested in making, possibly taking into account also the cost of performing an observation (Lindley, 1956; Howard, 1966). We may be interested in making an observation that minimizes our expected uncertainty about an event, or we may be interested in maximizing our expected utility; see, e.g. (Kjærulff and Madsen, 2008; Krause and Guestrin, 2009).

In this paper, we consider the use of a recently introduced notion, called the *Same-Decision Probability (SDP)*, as both a stopping criteria and a selection criteria for the purposes of more *robust* decision making. In short, the SDP is the probability of making the same decision even after knowing the values of a set of, as of yet, unobserved variables. (Darwiche and Choi, 2010). We first extend SDP, which was previously proposed for threshold-based decisions supported by Bayesian networks, to more general decision-making tasks, such as those typically supported

by influence diagrams. We next consider the potential value of SDP as a stopping criteria, via concrete examples, illustrating how SDP can quantify the stability of a decision in ways that are not evident when we consider beliefs and utilities alone. Next, we consider the potential value of SDP as a selection criteria, in terms of the gain in confidence (as opposed to information) than an observation may bring. Finally, we analyze the complexity of the generalized notion of SDP that we propose, showing that it is PPPP-complete. This result generalizes previous complexity results for SDP (Choi et al., 2012), but more importantly, applies to a broader class of VOI and reward functions.

2 Technical Preliminaries

We use standard notation for variables and their instantiations, where variables are denoted by upper case letters X and their instantiations by lower case letters x . Additionally, sets of variables are denoted by bold upper case letters \mathbf{X} and their instantiations by bold lower case letters \mathbf{x} . We assume that the state of the world is described over random variables \mathbf{X} , where the evidence $\mathbf{E} \subseteq \mathbf{X}$ includes **all** known variables, and where hidden variables $\mathbf{U} \subseteq \mathbf{X}$ include **all** unknown variables. By definition, $\mathbf{E} \cap \mathbf{U} = \emptyset$ and $\mathbf{E} \cup \mathbf{U} = \mathbf{X}$. We often discuss the ramifications of observing a subset of hidden variables $\mathbf{H} \subseteq \mathbf{U}$ on decision making. Furthermore, we use $D \in \mathbf{U}$ to denote the hypothesis variable that forms the basis for making a decision.¹

2.1 Same-Decision Probability

The same-decision probability (SDP) was initially defined in the context of threshold-based decisions (Darwiche and Choi, 2010), where a decision is made if the probability $Pr(d | \mathbf{e})$ reaches or surpasses a threshold T . Threshold-based decisions are common and can be found in troubleshooting (Heckerman et al., 1995), medical diagnosis (Pauker and Kassirer, 1980; van der Gaag and Coupé, 1999), anomaly detection (Kruegel et al., 2003), and fault diagnosis

¹One can extend the analysis to multiple hypothesis variables, but we focus here on the case of one hypothesis variable for simplicity.

(Lu and Przytula, 2006).

Although the SDP was defined in the context of threshold-based decisions, we extend the definition to a more general setting. In particular, we assume that \mathcal{F} is a function that outputs some decision given as input a distribution $Pr(D | \mathbf{e})$. SDP is thus defined as the probability that the same decision would be made if the hidden states of variables \mathbf{H} were known (Darwiche and Choi, 2010).

Definition 1. Given a decision function \mathcal{F} , hypothesis variable D , unobserved variables \mathbf{H} , and evidence \mathbf{e} , the **same-decision probability** (SDP) is defined as

$$SDP(\mathcal{F}, D, \mathbf{H}, \mathbf{e}) = \sum_{\mathbf{h}} [\mathcal{F}(Pr(D | \mathbf{h}, \mathbf{e}))]_{\mathbf{h}} Pr(\mathbf{h} | \mathbf{e}) \quad (1)$$

where $[\mathcal{F}(Pr(D | \mathbf{h}, \mathbf{e}))]_{\mathbf{h}}$ is an indicator function that is equal to 1 when $\mathcal{F}(Pr(D | \mathbf{h}, \mathbf{e})) = \mathcal{F}(Pr(D | \mathbf{e}))$, and equal to 0 otherwise.

Note that the original SDP definition assumed that we had a binary decision and would perform one decision if $Pr(d | \mathbf{e}) \geq T$ and the alternative decision otherwise (Darwiche and Choi, 2010).

2.2 Value of Information

We follow (Krause and Guestrin, 2009) by using a general definition of VOI based on reward functions. In particular, given an arbitrary reward function R ,² hypothesis D , hidden variables \mathbf{H} , and evidence \mathbf{e} , the VOI is defined as

$$\mathcal{V}(R, D, \mathbf{H}, \mathbf{e}) = \mathcal{ER}(R, D, \mathbf{H}, \mathbf{e}) - R(Pr(D | \mathbf{e})) \quad (2)$$

where

$$\mathcal{ER}(R, D, \mathbf{H}, \mathbf{e}) = \sum_{\mathbf{h}} R(Pr(D | \mathbf{h}, \mathbf{e})) Pr(\mathbf{h} | \mathbf{e}) \quad (3)$$

is the expected reward of observing variables \mathbf{H} and $R(Pr(D | \mathbf{e}))$ is the reward if we do

²We thoroughly discuss R in Section 5.

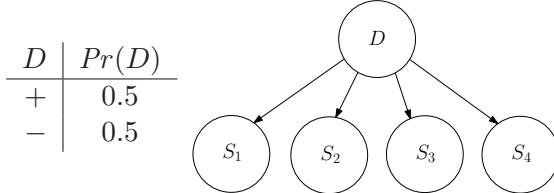


Figure 1: A Bayesian network for intrusion detection, with its CPTs given in Table 1.

not observe \mathbf{H} . Note that as in (Krause and Guestrin, 2009), VOI can refer to the expected reward as defined in Equation 3 since $R(Pr(D | \mathbf{e}))$ does not depend on variables \mathbf{H} .

For example, depending on the reward function used, VOI can be used to identify observations that best increase expected utility or reduce entropy (Kjærulff and Madsen, 2008). The mean-squared error and margins of confidence have also been used as reward functions (Krause and Guestrin, 2009).

3 SDP as a Stopping Criteria

We illustrate in this section the use of SDP as a stopping criteria in the context of threshold-based decisions and expected-utility decisions (i.e., influence diagrams).

3.1 Threshold-Based Decisions

Consider the sensor network in Figure 1, which may correspond to an intrusion detection application as discussed in (Kruegel et al., 2003). Here, the hypothesis variable is $D = \{+, -\}$ with $D = +$ implying an intrusion. Suppose we commit to a decision, and stop performing observations, when our belief in the event $D = +$ surpasses some threshold T , say $T = 0.55$. There are four sensors in this model, S_1, S_2, S_3 and S_4 , whose readings may affect this decision.

Consider the two following scenarios: (1) $S_1 = +$ and $S_2 = +$, and (2) $S_3 = +$ and $S_4 = +$. Since $Pr(D = + | S_1 = +, S_2 = +) = 0.60 > 0.55$ and $Pr(D = + | S_3 = +, S_4 = +) = 0.74 > 0.55$, it is clear that in both cases that the threshold has been crossed. We deem that no further observations are necessary based on our beliefs surpassing our threshold, as in (Kruegel et al., 2003; Lu and Przytula,

Table 1: CPTs for the network in Figure 1.

D	S_1	$Pr(S_1 D)$	D	S_3	$Pr(S_3 D)$
+	+	0.55	+	+	0.60
+	-	0.45	+	-	0.40
-	+	0.45	-	+	0.40
-	-	0.55	-	-	0.60

D	S_2	$Pr(S_2 D)$	D	S_4	$Pr(S_4 D)$
+	+	0.55	+	+	0.65
+	-	0.45	+	-	0.35
-	+	0.45	-	+	0.35
-	-	0.55	-	-	0.65

2006). Hence, when using thresholds as a stopping criteria, the two scenarios are identical.

From the viewpoint of SDP, however, these two scenarios are very different. In particular, the first scenario leads to an SDP of 52.97%. This means that there is a 47.03% chance that a different decision would be made if we were to further observe the two unobserved sensors S_3 and S_4 . The second scenario, however, leads to an SDP of 100%. That is, we would with certainty know that we would make the same decision, if we were to also observe the two unobserved sensors S_1 and S_2 : no matter what the readings of S_1 and S_2 could be, our beliefs in the event $D = +$ would always surpass our threshold 0.55. Indeed, as we can see in Table 1, the sensors S_1 and S_2 are not as strong as sensors S_3 and S_4 , and in this example, they are not strong enough to reverse our decision.

This example provides a clear illustration of the usefulness of the SDP as a stopping criteria. First, the SDP can pinpoint situations where further observations are unnecessary as they would never reverse the decision under consideration. Second, the SDP can also identify situations where the decision to be made is not robust, and is likely to change upon making further observations.

3.2 Expected-Utility Decisions

We now consider the use of SDP as a stopping criteria in the context of influence diagrams (Howard and Matheson, 1984).

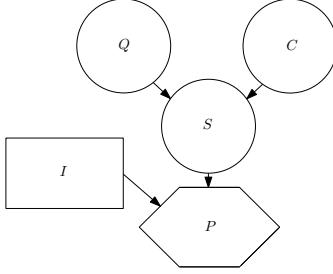


Figure 2: An influence diagram for an investment problem.

Consider the influence diagram in Figure 2, which consists of a Bayesian network with three variables (C , Q and S), a decision node I , and a utility node P that is a direct function of the utility function u . This influence diagram models an investment problem in which a venture capital firm is deciding whether to invest an amount of \$5 million in a tech startup ($I = T$) or to allow the money to collect interest in the bank ($I = F$). In this example, the profit of the investment (P) depends on the decision (I) and the success of the company (S), which in turn depends on two factors: (1) whether the existing competition is successful (C) and (2) whether the co-founders of the startup have a high quality, original idea (Q). Both C and Q are unobserved initially and independent of each other. Variable S is the hypothesis variable in this case and cannot be observed. Variables C and Q , however, can be observed for a price.

The goal here is to choose the decision $I = i$ with the maximum expected utility:

$$\mathcal{EU}(i | \mathbf{e}) = \sum_s Pr(s | \mathbf{e}) u(i, s)$$

where $u(i, s)$ is the utility of decision i given s , whether the company is successful or not.

Figures 3 and 4 contain two different parameterizations of the influence diagram in Figure 2. We will refer to these as different scenarios of the investment problem.

In both scenarios, given no evidence on variables C and Q , the best decision is $I = F$, with an expected utility of \$500K. A decision maker may commit to this decision or decide to observe variables C and Q , with the hope of finding a

Q	C	$Pr(S = T .)$
Q	$Pr(Q)$	
T	0.4	0.60
F	0.6	0.90
		0.20
		0.30

I	S	$u(I, S)$
I	$Pr(C)$	
T	0.6	$\$5 \times 10^6$
F	0.4	$-\$5 \times 10^6$
		$\$5 \times 10^5$
		$\$5 \times 10^5$

Figure 3: A parameterization of the influence diagram in Figure 2.

Q	C	$Pr(S = T .)$
Q	$Pr(Q)$	
T	0.1	0.05
F	0.9	0.98
		0.01
		0.05

I	S	$u(I, S)$
I	$Pr(C)$	
T	0.9	$\$7 \times 10^7$
F	0.1	$-\$5 \times 10^6$
		$\$5 \times 10^5$
		$\$5 \times 10^5$

Figure 4: A parameterization of the influence diagram in Figure 2.

better decision in light of the additional information. The classical stopping criteria here is to compute the *maximum* expected utility given that we observe variables C and Q (Heckerman et al., 1993; Kjærulff and Madsen, 2008):

$$\sum_{c,q} \max_i \mathcal{EU}(i | c, q) Pr(c, q)$$

In both scenarios, the maximum expected utility comes out to \$1,180K,³ showing that further observations may lead to a better decision of $I = T$, i.e. investing in the company.

Up to this point, the above two scenarios are indistinguishable from the viewpoint of classi-

³According to the formulation of (Krause and Guestrin, 2009), we have computed the VOI for variables C and Q using the reward function.

cal decision making tools. The SDP, however, finds these two scenarios very different. In particular, with respect to variables C and Q , the SDP is 60% in the first scenario and is 99% in the second scenario. That is, even though we stand to make a better decision of $I = T$ in both scenarios upon observing certain instantiations of C and Q , (at least with respect to utility), and even though the expected benefit from such observations is the same in both scenarios, it is very unlikely that we would change the current decision of $I = F$ in the second scenario in comparison to the first. Hence, given the additional information provided by the SDP, a decision maker may act quite differently in these two scenarios. Indeed, when we take a closer look at the second scenario, there is a state of the world that has very high utility (when $I = T$ and $S = T$). However, the chance of this state manifesting itself is extremely small (analogous to a lottery).

This illustrates the usefulness of SDP as a stopping criteria in the context of expected-utility decisions and influence diagrams. Namely, using SDP, we can distinguish between two very different scenarios, that are otherwise indistinguishable when we consider utilities alone.

4 SDP as a Selection Criteria

We now turn our attention to the use of SDP as a criteria for deciding which variables to observe next, assuming that some stopping criteria indicates that further observations are necessary.

Formally, our proposal is based on using VOI as the selection criteria (see Equation 2), while choosing the SDP as the reward function (see Equation 1). We next define SDP gain.

Definition 2. Given Definition 1 of SDP, the **SDP gain** of observing variables \mathbf{H} out of variables \mathbf{U} is defined as

$$\begin{aligned} \mathcal{G}(\mathbf{H}) = \sum_{\mathbf{h}} & SDP(\mathcal{F}, D, \mathbf{U} \setminus \mathbf{H}, \mathbf{he}) Pr(\mathbf{h}|\mathbf{e}) \\ & - SDP(\mathcal{F}, D, \mathbf{U}, \mathbf{e}). \end{aligned} \quad (4)$$

where \mathbf{he} denotes the joint instantiation of \mathbf{h} and \mathbf{e} .

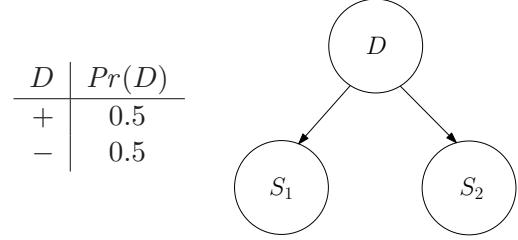


Figure 5: A Bayesian network with its CPTs given in Table 2.

Table 2: CPTs for the Bayesian network in Figure 5.

D	S_2	$Pr(S_2 D)$
+	+	0.75
+	o	0.2
+	-	0.05
-	+	0.05
-	o	0.2
-	-	0.75

The goal here is to observe those variables which, on average, will lead to the highest SDP. That is, we want to maximize the probability of making a decision that is unlikely to change even if we observe the remaining variables.

We will next provide an example of using SDP as a selection criteria, contrasting it with two other selection criteria: One based on reducing entropy of the hypothesis variable D , and another based on maximizing the margin between the first and second most likely states. While both criteria can be motivated as reducing uncertainty, we will show that both can indeed lead to less stable decisions in contrast to SDP.

The example is given by the Bayesian network in Figure 5, where D is the hypothesis variable and S_1/S_2 are sensors. A decision is triggered when $Pr(D = + | \mathbf{e}) \geq .80$, where evidence \mathbf{e} is over sensors S_1 and S_2 . With no observations (empty evidence \mathbf{e}), the SDP is 0.595, suggesting that further observations may be needed. Assuming a limited number of observations (Heckerman et al., 1995), and observing one variable at a time (Dittmer and Jensen, 1997), we need now to select the next variable to observe.

Note that maximizing VOI with negative entropy as the reward function amounts to maximizing mutual information (Cover and Thomas, 1991). The mutual information between variable D and sensor S_2 is 0.53 whereas the mutual information between D and sensor S_1 is 0.278. Hence, observing S_2 will reduce the entropy of D the most. In terms of margin of confidence, another reward function (Krause and Guestrin, 2009), observing S_2 will on average lead to a 0.7 margin between the $D = +$ and $D = -$, whereas observing S_1 will only lead to a 0.6 margin between the two states.

However, if we compute the corresponding SDP gains, $\mathcal{G}(S_1)$ and $\mathcal{G}(S_2)$, we find that observing S_1 will, on average, lead to improving the decision stability the most. In particular, observing S_1 would give us an SDP of either 1 or 0.81, resulting in an expected SDP of 0.905. Observing S_2 would give us an SDP of either 0.7625, 0.5, or 1, resulting in an expected SDP of 0.805. Therefore, $\mathcal{G}(S_1) = 0.31$ and $\mathcal{G}(S_2) = 0.21$. Hence, observing S_1 will on average allow us to make a decision that is less likely to change due to additional information.

Some intuition to why this is the case is that in threshold-based decisions, we make a decision solely based on whether $Pr(D | \mathbf{e})$ is above or below the threshold. Selection criteria such as entropy and margins of confidence will not consider the threshold. This example demonstrates the usefulness of SDP as a selection criteria for threshold-based decisions, as the SDP can be used to select observations that lead to more robust decisions.

5 Computational Complexity

The SDP was shown to be a PP^{PP}-complete problem in (Choi et al., 2012). The PP^{PP} class can be thought of as a counting variant of the NP^{PP} class, for which the MAP problem is complete (Park and Darwiche, 2004).

We show in this section that a general problem of computing expectations is also PP^{PP}-complete, with non-myopic VOI being an instance of such an expectation. We also show that the SDP is another instance of this com-

putation. Thus, the development of algorithms for SDP will be beneficial to problems in the complexity class PP^{PP}, which in turn benefits computing an assortment of expectations, including non-myopic VOI.

The proposed expectation computation is based on using a reward function R with some properties that we review next. In particular, the function R is assumed to map a probability distribution $Pr(D | \mathbf{e})$ to a numeric value. We also assume that the minimum l and maximum u of this range are polytime computable. These assumptions are not too limiting—for example, both entropy and utility can be expressed using reward functions that fall in this category (Krause and Guestrin, 2009).

We now consider the following computation of expectations.

D-EPT: Given reward function R , hypothesis variable D , unobserved variables \mathbf{H} , evidence \mathbf{e} , a real number N , and a distribution Pr induced by a Bayesian network over variables \mathbf{X} ,⁴ the **expectation decision problem** asks: Is

$$\mathbb{E} = \sum_{\mathbf{h}} R(Pr(D | \mathbf{h}, \mathbf{e})) Pr(\mathbf{h} | \mathbf{e})$$

greater than N ?

It should be clear that the SDP falls as a special case when the reward function R is the SDP indicator function (see Definition 1). For example, in (Choi et al., 2012), the decision function outputs one of two decisions depending on whether $Pr(d|\mathbf{e}) > T$ for some value d of D and some threshold T .

We now have the following theorems, with proofs in the Appendix.

Theorem 1. *D-EPT* is PP^{PP}-hard.

Theorem 2. *D-EPT* is in PP^{PP}.

This shows that **D-EPT** is PP^{PP}-complete. This also implies that computing the SDP is PP^{PP}-complete, as are other computational problems such as non-myopic VOI using a variety of reward functions.

⁴This proof also holds for influence diagrams constrained to have only one decision node.

6 Conclusion

In this paper, we extended the recently introduced notion of same decision probability (SDP), as a way to quantify the robustness of a decision, to a broader class of decision-making problems. Through concrete examples, we illustrated the usefulness of SDP as a stopping criterion, where SDP is capable of distinguishing scenarios that are otherwise indistinguishable based on thresholds or utilities alone. We further illustrated the usefulness of SDP as a selection criterion, in the terms of a confidence gain (in contrast to information gain). Finally, we provided a general analysis of the complexity of SDP, which extends itself to a broad class of functions formulated as expectations, including a general class of VOI computations.

7 Appendix

See (Kwisthout, 2009; Choi et al., 2012) for more on the complexity class PP^{PP} in the context of reasoning in Bayesian networks.

Proof of Theorem 1. We show **D-EPT** is PP^{PP} -hard by reduction from the following decision problem **D-SDP**, which corresponds to the originally proposed notion of same-decision probability for threshold-based decisions (Darwiche and Choi, 2010).

D-SDP: Given a decision based on probability $Pr(d | \mathbf{e})$ surpassing a threshold T , a set of unobserved variables \mathbf{H} , and a probability p , is the same-decision probability:

$$\sum_{\mathbf{h}} [Pr(d | \mathbf{h}, \mathbf{e}) \geq T] Pr(\mathbf{h} | \mathbf{e}) \quad (5)$$

greater than p ?

Here, $[.]$ denotes an indicator function which evaluates to 1 if the enclosed expression is satisfied, and 0 otherwise. **D-SDP** was shown to be PP^{PP} -complete in (Choi et al., 2012).

This same-decision probability corresponds to an expectation with respect to the distribution $Pr(\mathbf{H} | \mathbf{e})$, using the reward function:

$$R(Pr(D | \mathbf{h}, \mathbf{e})) = \begin{cases} 1 & \text{if } Pr(d | \mathbf{h}, \mathbf{e}) \geq T \\ 0 & \text{otherwise.} \end{cases}$$

Thus the same-decision probability is greater than T iff this expectation is greater than T . \square

Proof of Theorem 2. To show that **D-EPT** is in PP^{PP} , we provide a probabilistic polynomial-time algorithm, with access to a PP oracle, that answers the decision problem **D-EPT** correctly with probability greater than $\frac{1}{2}$. This proof generalizes and simplifies the proof given in (Choi et al., 2012) for **D-SDP**.

Consider the following probabilistic algorithm that determines if $\mathbb{E} > N$:

1. Sample a complete instantiation \mathbf{x} from the Bayesian network, with probability $Pr(\mathbf{x})$. We can do this in linear time, using forward sampling (Henrion, 1986).
2. If \mathbf{x} is compatible with \mathbf{e} , we can use a PP-oracle to compute $t = R(Pr(D | \mathbf{h}, \mathbf{e}))$. First, the reward function R can be computed in polynomial time, by definition. Second, $Pr(D | \mathbf{h}, \mathbf{e})$ can be computed using a PP-oracle, since the decision problem for marginals is PP-complete (Roth, 1996), and since $\text{P}^{\text{PP}} = \text{P}^{\#P}$.
3. Define a function $a(t) = \frac{1}{2} + \frac{1}{2} \frac{t-N}{u-l}$, which defines a probability used by our probabilistic algorithm to guess whether $\mathbb{E} > N$ (see Lemma 1).
4. Declare that $\mathbb{E} > N$ with probability:
 - $a(t)$ if \mathbf{x} is compatible with \mathbf{e} ;
 - $\frac{1}{2}$ if \mathbf{x} is not compatible with \mathbf{e} .

The probability of declaring $\mathbb{E} > N$ is:

$$r = \sum_{\mathbf{h}} a(t) Pr(\mathbf{h}, \mathbf{e}) + \frac{1}{2} (1 - Pr(\mathbf{e})) \quad (6)$$

which is greater than $\frac{1}{2}$ iff the following set of

equivalent statements hold:

$$\begin{aligned} \sum_{\mathbf{h}} a(t) Pr(\mathbf{h}, \mathbf{e}) &> \frac{Pr(\mathbf{e})}{2} \\ \sum_{\mathbf{h}} a(t) Pr(\mathbf{h} | \mathbf{e}) &> \frac{1}{2} \\ \sum_{\mathbf{h}} \left(\frac{1}{2} + \frac{1}{2} \frac{t-N}{u-l} \right) Pr(\mathbf{h} | \mathbf{e}) &> \frac{1}{2} \\ \sum_{\mathbf{h}} \left(\frac{1}{2} \frac{t-N}{u-l} \right) Pr(\mathbf{h} | \mathbf{e}) &> 0 \\ \sum_{\mathbf{h}} (t - N) Pr(\mathbf{h} | \mathbf{e}) &> 0 \\ \sum_{\mathbf{h}} R(Pr(D | \mathbf{h}, \mathbf{e})) Pr(\mathbf{h} | \mathbf{e}) &> N. \end{aligned}$$

Thus $r > \frac{1}{2}$ iff $\mathbb{E} > N$. \square

Lemma 1. *The function $a(t) = \frac{1}{2} + \frac{1}{2} \frac{t-N}{u-l}$ maps a reward t to a probability in $[0, 1]$.*

Proof. Values u and l are given, and denote upper and lower bounds on the reward t , but also the threshold N . Thus $\frac{t-N}{u-l}$ is in $[-1, 1]$. \square

Note that $a(t)$ denotes a probability used by our algorithm to declare whether $\mathbb{E} > N$, which is higher or lower depending on the value of the reward $t = R(Pr(D | \mathbf{h}, \mathbf{e}))$.

References

- Arthur Choi, Yexiang Xue, and Adnan Darwiche. 2012. Same-decision probability: A confidence measure for threshold-based decisions. *International Journal of Approximate Reasoning (IJAR)*.
- Thomas M. Cover and Joy A. Thomas. 1991. *Elements of information theory*. Wiley-Interscience, New York, NY, USA.
- Adnan Darwiche and Arthur Choi. 2010. Same-decision probability: A confidence measure for threshold-based decisions under noisy sensors. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*, pages 113–120. PGM.
- Soren Dittmer and Finn Jensen. 1997. Myopic value of information in influence diagrams. In *Proceedings of the Thirteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 142–149.
- Russell Greiner, Adam J. Grove, and Dan Roth. 1996. Learning active classifiers. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML96)*.
- David Heckerman, Eric Horvitz, and Blackford Middleton. 1993. An approximate nonmyopic computation for value of information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(3):292–298.
- David Heckerman, John S. Breese, and Koos Rommelse. 1995. Decision-theoretic troubleshooting. *Commun. ACM*, 38(3):49–57, March.
- Max Henrion. 1986. Propagating uncertainty in bayesian networks by probabilistic logic sampling. In *Uncertainty in Artificial Intelligence 2 Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, pages 149–163, Amsterdam, NL. Elsevier Science.
- Ronald A. Howard and James E. Matheson, editors. 1984. *Readings on the Principles and Applications of Decision Analysis*. Strategic Decision Group, Menlo Park, CA.
- Ronald A. Howard. 1966. Information value theory. *IEEE Trans. Systems Science and Cybernetics*, 2(1):22–26.
- Uffe B. Kjærulff and Anders L. Madsen. 2008. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Information Science and Statistics. Springer.
- Andreas Krause and Carlos Guestrin. 2009. Optimal value of information in graphical models. *Journal of Artificial Intelligence Research (JAIR)*, 35:557–591.
- Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. 2003. Bayesian Event Classification for Intrusion Detection. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, Las Vegas, NV USA, 12.
- Johan Kwisthout. 2009. *The Computational Complexity of Probabilistic Networks*. Ph.D. thesis, University of Utrecht.
- Dennis V. Lindley. 1956. On a measure of the information provided by an experiment. *Annals of Mathematical Statistics*, 27(4):986–1005.
- Tsai-Ching Lu and K. Wojtek Przytula. 2006. Focusing strategies for multiple fault diagnosis. In *FLAIRS Conference*, pages 842–847.
- James D. Park and Adnan Darwiche. 2004. Complexity Results and Approximation Strategies for MAP Explanations. *J. Artif. Intell. Res. (JAIR)*, 21:101–133.
- Steven G. Pauker and Jerome P. Kassirer. 1980. The threshold approach to clinical decision making. *N Engl J Med*, 302(20):1109–17.
- Dan Roth. 1996. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273 – 302.
- Linda C. van der Gaag and Veerle M. H. Coupé. 1999. Sensitive analysis for threshold decision making with bayesian belief networks. In *AI*IA*, pages 37–48.

Piecewise Linear Approximations of Nonlinear Deterministic Conditionals in Continuous Bayesian Networks

Barry R. Cobb

Virginia Military Institute
Lexington, Virginia, USA
cobbbr@vmi.edu

Prakash P. Shenoy

University of Kansas
Lawrence, Kansas, USA
pshenoy@ku.edu

Abstract

To enable inference in continuous Bayesian networks containing nonlinear deterministic conditional distributions, Cobb and Shenoy (2005) have proposed approximating nonlinear deterministic functions by piecewise linear ones. In this paper, we describe two principles and a heuristic for finding piecewise linear approximations of nonlinear functions. We illustrate our approach for some commonly used one- and two-dimensional nonlinear deterministic functions.

1 Introduction

This paper is concerned with inference in continuous Bayesian networks containing nonlinear deterministic conditional distributions for some continuous variables. In a BN, each variable is associated with a conditional probability distribution (or a conditional, in short) for each value of its parent variables. A conditional for a variable is said to be *deterministic* if the variances of the conditional are all zeroes (for all values of the variable's parents). If a continuous variable has a deterministic conditional, then the joint probability density function for all continuous variables does not exist, and this must be taken into account in a propagation algorithm for computing posterior marginals. Recently, Shenoy and West (2011a) have described an extension of the Shenoy-Shafer architecture for discrete BNs (Shenoy and Shafer, 1990), where deterministic conditionals for continuous variables are represented by Dirac delta functions (Dirac, 1927).

A major problem in inference in continuous BNs is marginalizing continuous variables, which involves integration. Often, there are no closed form solutions for the result of the integration, making representation of the intermediate functions difficult. We will refer to this as the *integration* problem.

One of the earliest non-Monte Carlo methods for inference in BNs with continuous variables was proposed by Lauritzen and Jensen (2001) for the special case where all continuous variables have conditional linear Gaussian (CLG) distributions. Since marginals of multivariate Gaussian distributions are multivariate Gaussian distributions whose parameters can be easily found from the parameters of the original distributions, this obviates the need to do integrations. However, the requirement that all continuous conditional distributions are CLG restricts the class of hybrid BNs that can be represented using this method.

Another method for dealing with the integration problem is the mixture of truncated exponentials (MTE) model proposed by Moral et al. (2001). The main idea here is to approximate conditional probability density functions (PDFs) by piecewise exponential functions, whose exponents are linear functions of the variables in the domain, and where the pieces are defined on hypercubes, i.e., intervals for each variable. Such functions are called MTEs, and this class of functions is closed under multiplication, addition, and integration, operations that are done in the propagation algorithm. Thus, the MTE method can be used for BNs that do not contain deterministic conditionals and any conditional distribution can be

used as long as they can be approximated by MTE functions.

Similar to the MTE method, Shenoy and West (2011b) have proposed another method called mixture of polynomials (MOP) to address the integration problem. The main idea is to approximate conditional PDFs by piecewise polynomials defined on hypercubes. In all other respects, the MOP method is similar in spirit to the MTE method.

Recently, Shenoy (2012) has proposed a generalization of the MOP function by allowing the pieces to be defined on regions called hyper-rhombuses, which are a generalization of hypercubes. One advantage of MOPs defined on hyper-rhombuses is that such functions are closed under transformations needed for multi-dimensional linear deterministic functions.

Cobb and Shenoy (2005) extend the applicability of MTE and MOP methods to continuous BNs containing nonlinear deterministic conditionals. The main idea is to approximate a nonlinear function by a piecewise linear (PL) function, and then apply the usual MTE or MOP method.

In this paper, we propose two principles and a heuristic for finding piecewise linear approximations of nonlinear functions, and illustrate it for an one-dimensional function $Y = X^2$, and a two-dimensional function $W = X \cdot Y$.

An outline of the remainder of the paper is as follows. In Section 2, we briefly define mixtures of polynomials functions. Also, we describe some numerical measures of goodness of an approximation of a PDF or cumulative distribution function (CDF). In Section 3, we describe two basic principles, and a heuristic, for finding piecewise linear approximations of a nonlinear functions in one and two dimensions, and we illustrate this technique for the functions $Y = X^2$ in the one-dimensional case, and $W = X \cdot Y$ for the two-dimensional case. Finally, in Section 4, we summarize our contributions and discuss some issues for further research.

2 Mixtures of Polynomials

In this section, we briefly define mixture of polynomials functions. For the remainder of the paper, all functions are assumed to equal zero in undefined regions.

The definition of mixture of polynomials given here is taken from (Shenoy, 2012).

An m -dimensional function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ is said to be a MOP function if

$$f(x_1, x_2, \dots, x_m) = \begin{cases} P_i(x_1, x_2, \dots, x_m) \\ \quad \text{for } (x_1, x_2, \dots, x_m) \in A_i, i = 1, \dots, k. \end{cases}$$

where $P_i(x_1, x_2, \dots, x_m)$ are multivariate polynomials in m variables for all i , and the regions A_i are disjoint and as follows. Suppose π is a permutation of $\{1, \dots, m\}$. Then each A_i is of the form:

$$\begin{aligned} l_{1i} &\leq x_{\pi(1)} \leq u_{1i}, \\ l_{2i}(x_{\pi(1)}) &\leq x_{\pi(2)} \leq u_{2i}(x_{\pi(1)}), \\ &\vdots \\ l_{mi}(x_{\pi(1)}, \dots, x_{\pi(m-1)}) &\leq x_{\pi(m)} \\ &\leq u_{mi}(x_{\pi(1)}, \dots, x_{\pi(m-1)}) \end{aligned} \quad (2.1)$$

where l_{1i} and u_{1i} are constants, and $l_{ji}(x_{\pi(1)}, \dots, x_{\pi(j-1)})$ and $u_{ji}(x_{\pi(1)}, \dots, x_{\pi(j-1)})$ are linear functions of $x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(j-1)}$ for $j = 2, \dots, m$, and $i = 1, \dots, k$. We will refer to the nature of the region described in Equation (2.1) as a *hyper-rhombus*.

A *hypercube* is a special case of a hyper-rhombus where $l_{1i}, \dots, l_{mi}, u_{1i}, \dots, u_{mi}$ are all constants.

Example 1. An example of a 2-piece, 3-degree MOP approximation $g_1(\cdot)$ of the standard normal PDF in 1-dimension is as follows:

$$g_1(x) = \begin{cases} 0.424 + 0.128x - 0.085x^2 - 0.028x^3 & \text{if } -3 < x < 0, \\ 0.424 - 0.128x - 0.085x^2 + 0.028x^3 & \text{if } 0 \leq x < 3 \end{cases} \quad (2.2)$$

$g_1(\cdot)$ was found using Lagrange interpolating polynomial with Chebyshev points (Shenoy, 2012).

The family of MOP functions is closed under multiplication, addition and integration, the operations that are done during propagation of potentials in the extended Shenoy-Shafer architecture for BNs. They are also closed under transformations needed for linear deterministic functions.

In this paper, we focus on the use of PL approximations in conjunction with MOP functions to facilitate inference in BNs. In many cases, the PL approximations can also be used with MTE functions. This is demonstrated in (Cobb and Shenoy, 2012).

2.1 Quality of MOP Approximations

In this section, we discuss some quantitative ways to measure the accuracy of a MOP approximation of PDFs.

We will measure the accuracy of a PDF with respect to another defined on the same domain by four different measures, the Kullback-Leibler (KL) divergence, maximum absolute deviation, absolute error in the mean, and absolute error in the variance.

If f is a PDF on the interval (a, b) , and g is a PDF that is an approximation of f such that $g(x) > 0$ for $x \in (a, b)$, then the *KL divergence* between f and g , denoted by $KL(f, g)$, is defined as

$$KL(f, g) = \int_a^b \ln\left(\frac{f(x)}{g(x)}\right) f(x) dx.$$

$KL(f, g) \geq 0$, and $KL(f, g) = 0$ if and only if $g(x) = f(x)$ for all $x \in (a, b)$. We do not know the semantics associated with the statistic $KL(f, g)$.

The *maximum absolute deviation* between f and g , denoted by $MAD(f, g)$, is given by:

$$MAD(f, g) = \sup\{|f(x) - g(x)| : a < x < b\}$$

One semantics associated with $MAD(f, g)$ is as follows. If we compute the probability of some interval $(c, d) \subseteq (a, b)$ by computing $\int_c^d g(x) dx$, then the error in this probability is bounded by $(d - c) \cdot MAD(f, g)$.

The maximum absolute deviation can also be applied to CDFs. Thus, if $F(\cdot)$ and $G(\cdot)$ are

the CDFs corresponding to $f(\cdot)$, and $g(\cdot)$, respectively, then the maximum absolute deviation between F and G , denoted by $MAD(F, G)$, is

$$MAD(F, G) = \sup\{|F(x) - G(x)| : a < x < b\}$$

The *absolute error of the mean*, denoted by $AEM(f, g)$, and the *absolute error of the variance*, denoted by $AEV(f, g)$, are given by:

$$AEM(f, g) = |E(f) - E(g)| \quad (2.3)$$

$$AEV(f, g) = |V(f) - V(g)| \quad (2.4)$$

where $E(\cdot)$ and $V(\cdot)$ denote the expected value and the variance of a PDF, respectively.

3 Finding PL Approximations of Nonlinear Functions

When we have nonlinear deterministic conditionals, Cobb and Shenoy (2005) propose approximating these nonlinear functions by piecewise linear (PL) ones. The family of MOP functions is closed under operation needed for linear deterministic functions.

There are many ways in which one can approximate a nonlinear function by a PL function. In this section, we describe two basic principles and a heuristic for minimizing the errors in the marginal distribution of the variable with the deterministic conditional represented by the PL approximation.

3.1 One-Dimensional Function $Y = X^2$

To illustrate PL approximations, consider a simple BN as follows: $X \sim N(0, 1)$, $Y = X^2$. The exact marginal distribution of Y is chi-square with 1 degree of freedom. We will use the 2-piece, 3-degree MOP $g_1(\cdot)$ defined in Equation (2.2) on the domain $(-3, 3)$, for the MOP approximation of the PDF of $N(0, 1)$.

3.1.1 Two Basic Principles

In constructing PL approximations, we will adhere to two basic principles. First, the domain of the marginal PDF of the variable with the deterministic conditional should remain unchanged. Thus, in the chi-square example, since the PDF of X is defined on the domain $(-3, 3)$,

and $Y = X^2$, the domain of Y is $(0, 9)$, and we need to ensure that any PL approximation of the function $Y = X^2$ results in the marginal PDF of Y on the domain $(0, 9)$.

Second, if the PDF of X is symmetric about some point, and the deterministic function is also symmetric about the same point, then we need to ensure that the PL approximation retains the symmetry. In the chi-square example, the PDF of X is symmetric about the point $X = 0$, and $Y = X^2$ is also symmetric about the point $X = 0$ on the domain $(-3, 3)$. Therefore, we need to ensure that the PL approximation is also symmetric about the point $X = 0$.

3.1.2 AIC-like Heuristic

In the statistics literature, Akaike's information criterion (AIC) (Akaike, 1974) is a heuristic for building statistical models from data. For example, in a multiple regression setting, if we have a data set with p explanatory variables and a response variable, we could always decrease the sum of squared errors in the model by using more explanatory variables. However, this could lead to over-fitting and lead to poor predictive performance. Thus, we need a measure that has a penalty factor for including more explanatory variables than is necessary. If we have a model with p explanatory variables, and $\hat{\sigma}^2$ is an estimate of σ^2 in the regression model, the AIC heuristic is to minimize $n \times \ln(\hat{\sigma}^2) + 2p$, where the $2p$ term acts like a penalty factor for using more explanatory variables than are necessary.

Our context here is slightly different from statistics. In statistics, we have data, and the true model is unknown. In our context, there is no data and the true model is known (the true model could be a nonlinear model estimated from data). However, there are some similarities. We could always decrease the error in the fit between the nonlinear function and the PL approximation by using more parameters (pieces), but doing so does not always guarantee that the error in the marginal distribution of the deterministic variable with the nonlinear function will be minimized. Making inferences with MOPs that have many pieces can be in-

tractable (Shenoy et al., 2011). For this reason, we need to keep the number of pieces as small as possible.

Suppose $f_X(x)$ denotes the PDF of X and suppose we approximate a non-linear deterministic function $Y = r(X)$ by a PL function, say $Y = r_1(X)$. The MSE of the PL approximation r_1 , denoted by $MSE(r_1)$, is given as follows:

$$MSE(r_1) = \int_{-\infty}^{\infty} f_X(x) (r(x) - r_1(x))^2 dx. \quad (3.1)$$

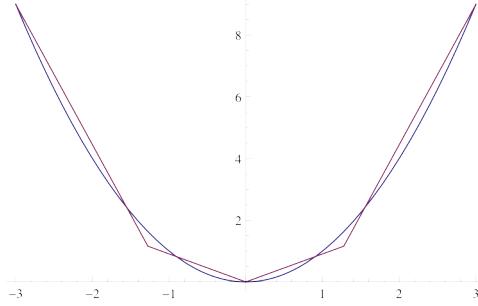
The AIC-like heuristic finds a PL approximation $Y = r_1(X)$ with p free parameters such that the $AIC(r_1) = \ln(MSE(r_1)) + p$ is minimized subject to the domain and symmetry principles.

3.1.3 Example

For the chi-square BN, the domain and symmetry principles require use of $(-3, 9)$, $(0, 0)$, and $(3, 9)$ knots. The knots are the points that are connected by straight lines to form the PL approximation. Suppose we wish to find a 4-piece PL approximation. Let (x_1, y_1) and $(-x_1, y_1)$ denote the two additional knots where $-3 < x_1 < 0$, and $0 < y_1 < 9$. Such a PL approximation would consist of 2 free parameters (where the parameters are x_1 and y_1). Solving for the minimum $MSE(r_1)$ with MOP $g_1(x)$ as the PDF of X results in the solution: $x_2 = -1.28$, $y_2 = 1.16$, minimum $MSE(r_1) = 0.043$, and $AIC(r_1) = -1.141$. The PL approximation $Y = r_1(X)$ is as follows (see Figure 1):

$$Y = \begin{cases} -4.66 - 4.55X & \text{if } -3 < X < -1.28 \\ -0.91X & \text{if } -1.28 \leq X < 0 \\ 0.91X & \text{if } 0 \leq X < 1.28 \\ -4.66 + 4.55X & \text{if } 1.28 \leq X < 3 \end{cases} \quad (3.2)$$

If we approximate $Y = X^2$ by a PL approximation $Y = r_2(X)$ with, say 6 pieces (4 free parameters), then the value of $MSE(r_2)$ is 0.0060, and the value of $AIC(r_2)$ is -1.124, which is higher than $AIC(r_1)$. Similarly, if we use an 8-piece approximation (6 free parameters), then the value of $MSE(r_3)$ is 0.002, and

Figure 1: $Y = r_1(X)$ overlaid on $Y = X^2$

the value of $AIC(r_3)$ is -0.421 , which is higher than $AIC(r_1)$ and $AIC(r_2)$. Thus, the AIC heuristic suggests a 2-piece PL approximation $Y = r_1(X)$. The accuracies of the marginal PDF of Y computed using MOP $g_1(x)$ for the PDF of X , and the three PL approximations r_1 , r_2 , and r_3 are shown in the table below (best values are shown in boldface). The model used as the exact PDF to calculate the goodness of fit statistics is the marginal PDF of Y found using $g_1(\cdot)$ and $Y = X^2$.

# pieces	4	6	8
p	2	4	6
MSE	0.043	0.006	0.002
AIC	-1.141	-1.124	-0.421
KL	0.250	0.153	0.110
MAD of PDF	35.332	35.092	34.713
MAD of CDF	0.164	0.133	0.101
AEM	0.059	0.189	0.168
AEV	0.065	0.106	0.186
CPU	4.774s	13.151s	22.932s

The MAD of the PDFs occurs near zero where the values of the actual PDF and the approximations are very large. The CPU row in the table above represents the combined time to obtain the PL approximation and calculate the goodness of fit statistics. The latter is an indication of the relative computing time that would be required to perform inference in a BN using these PL approximation with MOPs. All computations were made in Mathematica 8.0 on a computer with Intel Core 2 Duo processor (2.93 GHz) with 16 GB of memory.

The minimum AIC heuristic uses the information in the PDF of X as well as the nature

of the deterministic function $Y = g(X)$ to find a PL approximation. Its main disadvantage is that determining the knots of a minimum AIC PL approximation involves solving a nonlinear optimization problem. The more pieces there are in a PL approximation, the more variables there are in the nonlinear optimization problem, and more complex it is to solve the nonlinear optimization problem. However, we can often exploit special features of the PDF and the nonlinear function (such as the domain and symmetry principles) to minimize the number of variables in a optimization problem to make its solution tractable. Since the AIC-like heuristic includes a penalty for adding pieces to the PL approximation, the reduction in MSE associated with doing so may not be justified. In fact in this example, the AEM and AEV statistics for the resulting marginal distribution of interest are better when fewer pieces are used in the PL approximation. As expected, computation time increases with the number of pieces in the PL approximation.

3.2 Multi-Dimensional Function

$$W = X \cdot Y$$

For multi-dimensional nonlinear functions, we can use the same two basic principles and the minimum AIC-like heuristic as for the one-dimensional case.

Consider this example: $X \sim N(5, 0.5^2)$, $Y \sim N(15, 4^2)$, and $W = r(X, Y) = X \cdot Y$. We construct a 2-piece, 3-degree MOP $g_X(x) = g_1\left(\frac{x-5}{0.5}\right)/0.5$ of the PDF of X on the domain $(3.5, 6.5)$, and a 2-piece, 3-degree MOP $g_Y(y) = g_1\left(\frac{y-15}{4}\right)/4$ of the PDF of Y on the domain $(3, 27)$ (here $g_1(\cdot)$ is the 2-piece, 3-degree MOP approximation of the standard normal PDF on the domain $(-3, 3)$ as described in Equation 2.2).

Using these two MOP approximations of the PDFs of X and Y , we can find an “exact” marginal PDF of W as follows:

$$g_W(w) = \int_{-\infty}^{\infty} g_X(x) \left(\int_{-\infty}^{\infty} g_Y(y) \delta(w - x \cdot y) dy \right) dx, \quad (3.3)$$

where δ is the Dirac delta function (Shenoy

and West, 2011a). $\delta(w - x \cdot y)$ represents the conditional distribution of W given $X = x$ and $Y = y$. $g_W(\cdot)$ is not a MOP, but we do have a representation of it, can graph it, and can compute its mean ($E(g_W) = 75$) and variance ($V(g_W) = 458.96$). Unfortunately, we cannot compute the CDF corresponding to $g_W(\cdot)$. So we do not report any *MAD* for the CDFs statistics.

Suppose we wish to find a 2-piece PL approximation of $W = X \cdot Y$. The domain of the joint distribution of X and Y is a rectangle $(3.5 < X < 6.5) \times (3 < Y < 27)$. The exact domain of W is $(10.5, 175.5)$. Because g_X is symmetric about the axis $X = 5$, and g_Y is symmetric about the axis $Y = 15$, there are several ways one can find a two-piece region using the symmetry principle. The PDF of X is symmetric about the line $X = 5$, while the PDF of Y is symmetric about the line $Y = 15$. The slope and intercept of the line connecting the points $(3.5, 3)$ and $(6.5, 27)$ are 8 and -25 , respectively, so the joint PDF of (X, Y) is symmetric about the line $Y = 8X - 25$. The joint PDF of (X, Y) is similarly symmetric about the line $Y = -8X + 55$. There is no symmetry in the function $W = X \cdot Y$ about any axis. Thus, we can divide the domain vertically using the hyperplane $X = 5$ or horizontally using $Y = 15$ or diagonally using $Y = 8X - 25$ or $Y = -8X + 55$. The best approximation (lowest AIC score) obtained was by dividing the domain of X and Y vertically using the hyperplane $X = 5$. We conjecture that this works better than dividing the rectangle vertically since X has a smaller variance than Y .

Next, we need to find a PL approximation for $r(X, Y) = X \cdot Y$ in each of the two rectangles that satisfies the domain principle. The smallest value of $W = X \cdot Y$ is 10.5 at the point $(X, Y) = (3.5, 3)$, and the largest value of W is 175.5 at the point $(X, Y) = (6.5, 27)$. For the first rectangle, $3.5 < X < 5, 3 < Y < 27$, consider a PL approximation $r_{11}(X, Y) = a_1X + b_1Y + c_1$, where a_1, b_1, c_1 are constants. One way to satisfy the lower bound domain constraint is by selecting the PL approximation r_1 such that $r_{11}(3.5, 3) = 10.5$. Thus, we

can eliminate one of the three parameters, e.g., $c_1 = 10.5 - 3.5a_1 - 3b_1$. To find values of the remaining two parameters, we solve an optimization problem as follows:

Find a_1, b_1, c_1 so as to

$$\text{Minimize } \int_{3.5}^5 g_X(x) \left(\int_3^{27} (r(x, y) - r_{11}(x, y))^2 g_Y(y) dy \right) dx \quad (3.4)$$

subject to : $c_1 = 10.5 - 3.5a_1 - 3b_1$

For the second rectangle $5 \leq X < 6.5, 3 < Y < 27$, consider a PL approximation $r_{12}(X, Y) = a_2X + b_2Y + c_2$. To satisfy the upper domain constraint, we impose the constraint $r_{12}(6.5, 27) = 175.5$, i.e., $c_2 = 175.5 - 6.5a_2 - 27b_2$. To find values of the remaining two parameters, we solve the optimization problem:

Find a_2, b_2, c_2 so as to

$$\text{Minimize } \int_5^{6.5} g_X(x) \left(\int_3^{27} (r(x, y) - r_{12}(x, y))^2 g_Y(y) dy \right) dx \quad (3.5)$$

subject to : $c_2 = 175.5 - 6.5a_2 - 27b_2$, and

$$5a_2 + 3b_2 + c_2 \geq 10.5$$

The logic behind the second constraint in (3.5) is that assuming $a_2 \geq 0$ and $b_2 \geq 0$ (which are implicitly satisfied in minimizing MSE), the smallest value of $W = r_{12}(X, Y)$ is at the point $(X, Y) = (5, 3)$. Thus, in order to satisfy the domain principle, we need to ensure that $r_{12}(5, 3) \geq 10.5$. This constraint is binding at the optimal solution. Solving the optimization problems in (3.4) and (3.5), we obtain a PL approximation r_1 as follows:

$$r_1(X, Y) = \begin{cases} 8.15X + 4.18Y - 30.55 & \text{if } X < 5 \\ 25.51X + 5.28Y - 132.89 & \text{if } X \geq 5 \end{cases} \quad (3.6)$$

The total MSE for $r_1(X, Y)$ when compared to $r(X, Y)$ using PDFs $g_X(x)$ and $g_Y(y)$ is 14.98. Since we have 4 free parameters (a_1, b_1, a_2, b_2) in the PL approximation $r_1(X, Y)$, the AIC value is $AIC(r_1) = 6.71$.

Let $g_{W_1}(\cdot)$ denote the marginal PDF of W computed using $g_X(x)$, $g_Y(y)$, and $\delta(w - r_1(x, y))$. $g_{W_1}(\cdot)$ is computed as a 13-piece, 7-degree MOP on the domain $(10.5, 175.5)$. A

graph of $g_{W_1}(\cdot)$ overlaid on the graph of $g_W(\cdot)$ is shown in Figure 2.

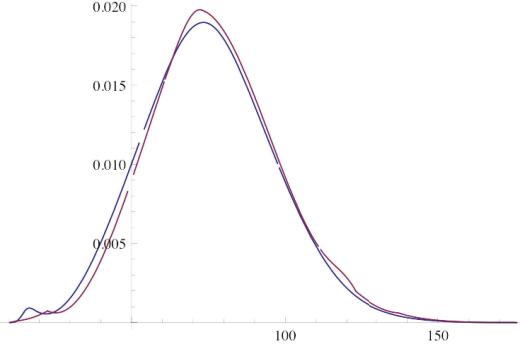


Figure 2: $g_{W_1}(\cdot)$ overlaid on $g_W(\cdot)$

The goodness of fit statistics of g_{W_1} compared to g_W are as follows:

Errors	Values
$KL(g_W, g_{W_1})$	0.0069
$MAD(g_W, g_{W_1})$	0.0012
$AEM(g_W, g_{W_1})$	1.8074
$AEV(g_W, g_{W_1})$	12.7308

One way to reduce the AIC value for the PL approximation is to reduce its number of parameters. In solving the optimization problems (3.4) and (3.5), if we add the constraints $c_1 = -a_1 \cdot b_1$ and $c_2 = -a_2 \cdot b_2$, we obtain a PL approximation r_2 as follows:

$$r_2(X, Y) = \begin{cases} 3.00X + 4.60Y - 13.81 & \text{if } X < 5 \\ 27.00X + 5.19Y - 140.06 & \text{if } X \geq 5 \end{cases} \quad (3.7)$$

The approximation $W = r_2(X, Y)$ has a MSE of 18.10, compared to MSE of 14.98 for $W = r_1(X, Y)$. Notice that the approximation $W = r_2(X, Y)$ has only 2 free parameters (compared to 4 for $W = r_1(X, Y)$). The corresponding values of the AIC heuristic is $AIC(r_2) = 4.90$. Let $g_{W_2}(\cdot)$ denote the marginal PDF of W computed using $g_X(x)$, $g_Y(y)$, and $\delta(w - r_2(x, y))$. $g_{W_2}(\cdot)$ is computed as a 13-piece, 7-degree MOP on the domain (10.5, 175.5). A graph of $g_{W_2}(\cdot)$ overlaid on the graph of $g_W(\cdot)$ is shown in Figure 3.

The goodness of fit statistics of g_{W_2} compared to g_W are as follows:

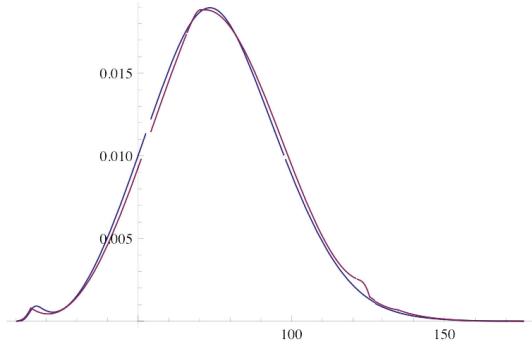


Figure 3: $g_{W_2}(\cdot)$ overlaid on $g_W(\cdot)$

Errors	Values
$KL(g_W, g_{W_2})$	0.0023
$MAD(g_W, g_{W_2})$	0.0008
$AEM(g_W, g_{W_2})$	1.2620
$AEV(g_W, g_{W_2})$	10.5493

Comparing these statistics with those obtained without the constraints $c = -a \cdot b$, we see that even though the MSE of r_2 is higher, all goodness of fit statistics for g_{W_2} (computed using r_2) are better than the corresponding ones for g_{W_1} (computed using r_1). This is probably because the AIC value of r_2 (4.90) is lower than the AIC value of r_1 (6.71). The AIC value of r_2 is lower than the AIC value of r_1 since r_2 has 2 less parameters than r_1 .

4 Summary and Conclusions

This paper is concerned with inference in BNs containing nonlinear deterministic conditionals using MOPs. The family of MOP functions is not closed under operations needed for inference with nonlinear deterministic conditionals, but is closed for inference with linear deterministic conditionals. Cobb and Shenoy (2005) suggest approximating nonlinear deterministic conditionals by piecewise linear ones. However, there are many ways of finding such approximations, and a very naïve heuristic was used in (Cobb and Shenoy, 2005), which examined only 1-dimensional nonlinear functions.

In this paper, we describe a principled approach to finding PL approximations of nonlinear functions. The domain principle ensures that the domain of the PL approximation should be exactly the same as in the nonlinear

case, and the symmetry principle states that the approximation should retain symmetry of the nonlinear function (if any), and the symmetry of the PDFs of the parent variables (if any). An AIC-like heuristic for finding PL approximation is described.

Using these two principles and the AIC-like heuristic, PL approximations of some commonly used nonlinear functions are described. For the nonlinear functions $Y = X^2$ and $W = X \cdot Y$, we find the marginal of the variable with the nonlinear deterministic conditional using PL approximations, and compare it with the marginal found using the exact nonlinear function, and estimate the errors in the marginals using MOP approximations of PDFs.

Cobb and Shenoy (2012) use the principles and heuristic in this paper to find PL approximations of $Y = e^X$ and $W = 3X/Y$, and also solve two small hybrid Bayesian networks that contain nonlinear deterministic conditionals. The use of MTE functions in combination with PL approximations to nonlinear deterministic conditionals is also demonstrated.

Acknowledgments

This research has been funded in part by the Spanish Ministry of Economy and Competitiveness through project TIN2010-20900-C04-01, and by the European Regional Development Funds (FEDER).

References

- H. Akaike. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.
- B. R. Cobb and P. P. Shenoy. 2005. Nonlinear deterministic relationships in Bayesian networks. In L. Godo, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Lecture Notes in Artificial Intelligence 3571, pages 27–38. Springer, Berlin.
- B. R. Cobb and P. P. Shenoy. 2012. Modeling nonlinear deterministic conditional distributions in hybrid Bayesian networks. Working Paper 328, University of Kansas, School of Business, Lawrence, KS.
- P. A. M. Dirac. 1927. The physical interpretation of the quantum dynamics. *Proceedings of the Royal Society of London, Series A*, 113(765):621–641.
- S. L. Lauritzen and F. Jensen. 2001. Stable local computation with conditional Gaussian distributions. *Statistics & Computing*, 11(2):191–203.
- S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In S. Benferhat and P. Besnard, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Lecture Notes in Artificial Intelligence 2143, pages 156–167. Springer, Berlin.
- P. P. Shenoy and G. Shafer. 1990. Axioms for probability and belief-function propagation. In R. D. Shachter, T. Levitt, J. F. Lemmer, and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence 4*, pages 169–198. North-Holland.
- P. P. Shenoy and J. C. West. 2011a. Extended Shenoy-Shafer architecture for inference in hybrid Bayesian networks with deterministic conditionals. *International Journal of Approximate Reasoning*, 52(6):805–818.
- P. P. Shenoy and J. C. West. 2011b. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52(5):641–657.
- P. P. Shenoy, R. Rumí, and A. Salmerón. 2011. Some practical issues in inference in hybrid Bayesian networks with deterministic conditionals. In S. Ventura, A. Abraham, K. Cios, C. Romero, F. Marcelloni, J. M. Benitez, and E. Gibaja, editors, *Proceedings of the 2011 Eleventh International Conference on Intelligent Systems Design and Applications*, pages 605–610. IEEE Research Publishing Services, Piscataway, NJ.
- P. P. Shenoy. 2012. Two issues in using mixtures of polynomials for inference in hybrid Bayesian networks. *International Journal of Approximate Reasoning*, 53(5):847–866.

Approximating the Distribution of a Sum of Log-normal Random Variables

Barry R. Cobb

Virginia Military Institute
Lexington, VA, USA
cobbbr@vmi.edu

Rafael Rumí

Antonio Salmerón
Universidad de Almería, Spain
rrumi,antonio.salmeron@ual.es

Abstract

This paper introduces a process for estimating the distribution of a sum of independent and identically distributed log-normal random variables (RVs). The procedure involves using the Fenton-Wilkinson method to estimate the parameters for a single log-normal distribution that approximates the sum of log-normal RVs. Once these parameters are determined, a mixture of truncated exponentials (MTE) function is determined to approximate this distribution. The MTE parameters are stated as polynomial functions of the log-normal scale parameter. Applications to inventory management are presented that demonstrate the usefulness of the MTE approximation.

1 Introduction

Finding the probability density function (PDF) for a sum of log-normally distributed random variables (RVs) is an important problem in business and telecommunications (Beaulieu et al., 1995). This paper proposes a tractable approximation to the PDF for a sum of log-normal RVs that can be utilized in Bayesian networks (BNs) and influence diagrams (IDs).

Consider the following business application in inventory management. Suppose the independent and identically distributed (i.i.d.) values of customer demand for a business, X_ℓ , in each period $\ell = 1, \dots, L$, are log-normally distributed, i.e. $X_\ell \sim LN(\mu_{X_\ell}, \sigma_{X_\ell}^2)$. This means that total customer demand over the L periods, X , is determined as the following sum of i.i.d. RVs:

$$X = X_1 + X_2 + X_3 + \dots + X_L . \quad (1)$$

The value L represents the fixed lead time between the placement and arrival of an inventory order. The business needs to establish order quantity and reorder point policies that minimize inventory costs. Finding a tractable approximation to the distribution for X is important to solving this problem.

Fenton (1960) and Schwartz and Yeh (1982) estimate the PDF for a sum of log-normal RVs

using another log-normal PDF with the same mean and variance. The Fenton approximation (sometimes referred to as the *Fenton-Wilkinson* (FW) method) is simpler to apply, and for a wide range of log-normal parameters has been shown to be reasonably accurate in comparison to the Schwartz-Yeh (SY) method (Beaulieu et al., 1995).

While the FW method produces a PDF that in some cases is a good approximation to the PDF for a sum of log-normals, this still does not result in a tractable representation that can be incorporated in BNs and IDs. This is because such models require a functional form that can be combined with other functions, then integrated in closed-form. This paper introduces a new mixtures of truncated exponentials (MTE) approximation to the log-normal PDF. MTE functions were introduced by Moral et al. (2001) to facilitate closed-form mathematical calculations in BNs.

One technique (Cobb et al., 2006) suggested for approximating a log-normal PDF with an MTE function requires a nonlinear optimization problem to be solved for each possible combination of μ and σ^2 . In this paper, an approximation is introduced that calculates the MTE parameters as polynomial functions of only the scale parameter, σ , for a range of values needed

to model the inventory application. This reduces the number of tabular inputs required to efficiently incorporate the FW method into an inventory management system using BNs or IDs.

The next section gives definitions and notation. Section 3 describes the MTE approximation. Section 4 applies the MTE approximation to inventory management examples and compares the results to those obtained with other techniques. The final section summarizes and gives suggestions for future research.

2 Definitions

Variables in this paper will be denoted by capital letters, e.g., X, Y, Z , with specific values shown in lower-case, e.g., x, y, z . The state space of X is denoted by Ω_X . The expressions $\exp(x)$ and e^x are used interchangeably.

2.1 Log-normal Distribution

A RV X is log-normal, i.e. $X \sim LN(\mu, \sigma^2)$, if and only if $\ln(X) \sim N(\mu, \sigma^2)$. A log-normal RV has the PDF

$$f_X(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(\ln x - \mu)^2}{2\sigma^2}\right), \quad x > 0,$$

for any $\sigma^2 > 0$. The expected value of X is $E(X) = \exp(\mu + 0.5\sigma^2)$ and the variance of X is $Var(X) = (\exp(\sigma^2) - 1)\exp(2\mu + \sigma^2)$.

The mode of $f_X(x)$ is $m = \exp(\mu - \sigma^2)$ and the inflection points to the right and left of the mode are $d^\pm = \exp\left(\frac{1}{2}(2\mu - 3\sigma^2 \pm \sigma\sqrt{4 + \sigma^2})\right)$.

If $X \sim LN(\mu, \sigma^2)$, then $aX \sim LN(\mu + \ln(a), \sigma^2)$ where $a > 0$. Conveniently, then, we can find a PDF for a $X \sim LN(\mu, \sigma^2)$ as a convolution of $Y \sim LN(0, \sigma^2)$ as follows:

$$f_X(x) = (1/\exp(\mu)) \cdot f_Y((1/\exp(\mu)) \cdot x) \quad (2)$$

2.2 Mixtures of Truncated Exponentials

The functional form of the log-normal PDF does not permit integration in closed-form. This makes this PDF difficult to incorporate into BNs and IDs. One approach that can allow log-normal RVs to be included in these models is to

approximate log-normal PDFs in the BN or ID with MTE potentials.

MTE Potential (Moral et al., 2001). Let X be a continuous chance variable. Given a partition $\Omega_1, \dots, \Omega_n$ that divides Ω_X into hypercubes, an n -piece, m -term MTE potential $\phi : \Omega_X \mapsto \mathcal{R}^+$ has components

$$\phi_h(x) = a_{1h} + \sum_{i=1}^m a_{2i,h} \exp(a_{2i+1,h} \cdot x)$$

for $h = 1, \dots, n$, where $a_{jh}, j = 1, \dots, 2m+1$ are real numbers. Thus, an n -piece, m -term MTE potential requires $2mn + n$ parameters.

MTE potentials can approximate both PDFs and utility functions. The optimization procedures outlined by Cobb et al. (2006) and Langseth et al. (2012) can be used to determine the parameters (the values a_{jh}) required to approximate PDFs with MTEs.

2.3 Fenton-Wilkinson (FW) Approximation

Consider the sum of L i.i.d. log-normal RVs, X , as specified in (1) where each $X_\ell \sim LN(\mu_{X_\ell}, \sigma_{X_\ell}^2)$ with the expected value and variance described in Section 2.1. The expected value and variance of X are $E(X) = L \cdot E(X_\ell)$ and $Var(X) = L \cdot Var(X_\ell)$. The FW approximation is a log-normal PDF with parameters μ_X and σ_X^2 such that $\exp(\mu_X + 0.5\sigma_X^2) = L \cdot E(X_\ell)$ and

$$(\exp(\sigma_X^2) - 1) \cdot \exp(2\mu_X + \sigma_X^2) = L \cdot Var(X_\ell).$$

Solving for μ_X and σ_X^2 gives

$$\sigma_X^2 = \ln((\exp(\sigma_{X_\ell}^2) - 1)/L + 1) \quad (3)$$

and

$$\mu_X = \ln(L \cdot \exp(\mu_{X_\ell})) + 0.5(\sigma_{X_\ell}^2 - \sigma_X^2). \quad (4)$$

Example 1. Suppose $X_\ell, \ell = 1, \dots, 5$ are distributed as $X_\ell \sim LN(0.69, 1.07^2)$. The sum, X , of these i.i.d. RVs has expected value $E(X) = 5 \cdot \exp(0.694 + 0.5 \cdot 1.074^2) = 17.8$ and $Var(X) = 5 \cdot Var(X_\ell) = 137.5$. Applying the FW approximation entails using the $LN(2.7, 0.6^2)$ distribution as an estimated PDF

for X . Note that this distribution has an expected value of $\exp(2.7 + 0.5 \cdot 0.6^2) = 17.8$ and a variance of $(e^{0.6^2} - 1)(e^{2 \cdot 2.7} e^{0.6^2}) = 137.5$.

3 MTE Approximation to the Log-normal Distribution

This section describes some enhancements to the MTE approximation to the log-normal PDF detailed by Cobb et al. (2006). Specifically, the previous model required the MTE parameters a_{jh} to be determined for each combination of μ and σ^2 . However, by applying the convolution in (2), we only need to estimate an MTE density function for each value of σ .

Note from (3) that σ_X^2 will be smaller than $\sigma_{X_\ell}^2$ and will decrease as L increases. For instance, for $\sigma_{X_\ell}^2 = 2$, $\sigma_X^2 < 1$ if $L \geq 4$. The FW approximation has been shown to be adequate when $\sigma_{X_\ell}^2 < 2$ (Beaulieu et al., 1995). Additionally, for inventory management problems, such as the one mentioned earlier, Das (1983) has observed that $\sigma_X^2 < 1$ normally holds. Thus, focusing on a range for σ_X in the interval $[0.05, 1]$ will be adequate for the problems in Section 4.

3.1 Regions for σ and Domain Splits

Four regions for the log-normal parameter σ are established as: $R_1 : [0.05, 0.1]$, $R_2 : [0.1, 0.2]$, $R_3 : [0.2, 0.63]$, and $R_4 : [0.63, 1]$. The values $\sigma = 0.1$ and $\sigma = 0.2$ are chosen because the shape of the log-normal PDF changes rapidly for σ values near zero. The value $\sigma = 0.63$ is chosen as the value to divide R_3 and R_4 because it is the point where $d^- \approx \exp(-2\sigma)$, thus this value determines the domains of two pieces of the MTE potential, as shown below.

Suppose we have a RV $Y \sim LN(0, \sigma^2)$. We will use a 5-piece, 2-term MTE approximation to the PDF for Y . The domains of the five pieces are determined using:

$$\begin{aligned} y_0 &: \exp(-3\sigma) & y_1 &: \text{Max}\{d^-, \exp(-2\sigma)\} \\ y_2 &: m = \exp(-\sigma^2) & y_3 &: d^+ \\ y_4 &: \exp(\sigma) & y_5 &: \exp(3\sigma) \end{aligned}$$

3.2 MTE Specification

The un-normalized 5-piece, 2-term MTE approximation to the $LN(0, \sigma^2)$ PDF is

$$\hat{f}_Y^{(u)}(y) = \begin{cases} \hat{a}_{1h}(\sigma) + \\ \sum_{i=1}^2 \hat{a}_{2i,h}(\sigma) \exp(\hat{a}_{2i+1,h}(\sigma)(y - m(\sigma))) \\ \quad \text{if } y_{h-1} \leq y < y_h \end{cases} \quad (5)$$

for $h = 1, \dots, 5$. All functions are assumed to equal zero in undefined regions. By adding pieces, or by fitting more terms in the first and fifth pieces, the domain of the MTE function could be extended, at the expense of additional computational burden in making calculations with the resulting distribution (Cobb et al., 2006; Romero et al., 2006). Any selection of number of pieces and terms in an MTE function involves some trade-off between accuracy and computational burden.

3.3 Functional Representation of MTE Parameters

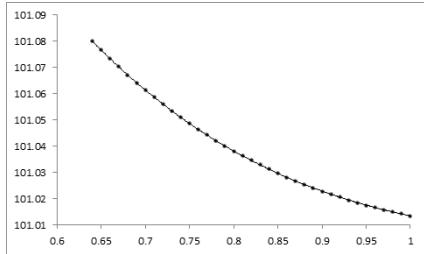
To reduce the tabular inputs required to completely define the family of MTE approximations to the $LN(0, \sigma^2)$ PDFs, we use approximate parameters \hat{a}_{jh} . These parameters are determined using the polynomial functions

$$\hat{a}_{jh}(\sigma) = \sum_{k=1}^5 \hat{c}_{kjh}^{(r)} \sigma^{k-1} \quad (6)$$

for $h = 1, \dots, 5$ and $r = 1, \dots, 4$. The value of r is determined according to the region – R_1 , R_2 , R_3 , or R_4 – where σ is located. Thus, to define the parameters for a 5-piece, 2-term MTE distribution where σ is located in any of these four regions requires a table of $4 \cdot 5 \cdot (2 \cdot 2 \cdot 5 + 5) = 500$ constants, $\hat{c}_{kjh}^{(r)}$.

To estimate the constants $\hat{c}_{kjh}^{(r)}$, we use the Cobb et al. (2006) nonlinear optimization procedure to find the “exact” MTE parameters a_{jh} for $0.05 \leq \sigma \leq 1$ in increments of 0.01 for σ . Linear regression is then used to estimate the model in (6). As an example, Figure 1 shows a plot of the actual MTE parameter a_{24} for values $0.63 \leq \sigma \leq 1$, as determined using the nonlinear optimization procedure. The equation

$$\begin{aligned} \hat{a}_{24}(\sigma) = & 101.8201 - 2.5357\sigma \\ & + 3.2139\sigma^2 - 1.9887\sigma^3 + 0.5033\sigma^4 \end{aligned}$$

Figure 1: Plots of a_{24} (dots) and \hat{a}_{24} (line).

is estimated using least squares regression and can be used to find one of the MTE parameters required to approximate any log-normal PDF with $\sigma \in R_4$. This equation fits the actual data ($R^2 = 1$ to four decimal places) very closely and is represented by the curve connecting the dots in Figure 1.

Once the constants $\hat{c}_{kjh}^{(r)}$ are stored, no further use of the nonlinear optimization procedure is required and a log-normal PDF with any μ value and any $\sigma \in [0.05, 1]$ can be approximated.

3.4 Summarized MTE Approximation Process

Ultimately, we want to approximate the distribution for $X = X_1 + \cdots + X_L$ where X_ℓ , $\ell = 1, \dots, L$ are i.i.d. $LN(\mu_{X_\ell}, \sigma_{X_\ell}^2)$ RVs. Given the formulation we have described for approximating the $LN(0, \sigma^2)$ PDF, the process can be summarized as follows:

1. Calculate the FW parameters σ_X^2 and μ_X from expressions (3) and (4).
2. Calculate the MTE parameters $\hat{a}_{jh}(\sigma_X)$ using equation (6).
3. Construct an un-normalized version, $\hat{f}_Y^{(u)}(y)$, of the MTE approximation to the PDF for a RV Y that has the $LN(0, \sigma_X^2)$ distribution using (5). Although the “exact” parameters $a_{jh}(\sigma)$ were determined for a PDF that integrated to 1, because the $\hat{a}_{jh}(\sigma)$ are approximate parameters, the result of expression (5) is not guaranteed to integrate to 1.
4. Normalize the PDF for Y by calculating $k_Y = \int_{-\infty}^{\infty} \hat{f}_Y^{(u)}(y) dy$ and finding $\hat{f}_Y(y) = k_Y^{-1} \cdot \hat{f}_Y^{(u)}(y)$.

5. Use the convolution operation in (2) to create the normalized PDF $\hat{f}_X(x)$ for X .

Example 2. Continuing from Example 1, to construct an MTE approximation to the $LN(2.7, 0.6^2)$ PDF for X , we first need to build the MTE approximation to the $LN(0, 0.6^2)$ PDF. Table 1 shows the constants $c_{kj1}^{(3)}$ needed to determine the parameters for the first (of five) pieces of this MTE potential. The second row shows σ^{k-1} , $k = 1, \dots, 5$, while the other rows show the constants needed to calculate each of the five parameters \hat{a}_{j1} . These parameters are calculated in the last column as the sumproduct of the σ^{k-1} row and the corresponding row of $\hat{c}_{kj1}^{(3)}$ s.

The un-normalized MTE approximation to the $LN(0, 0.6^2)$ (with mode $m = 0.698$) PDF is

$$\begin{cases} \hat{f}_Y^{(u)}(y) = \\ -0.098 + 87.823e^{10.208(y-m)} \\ -113.664e^{11.609(y-m)} & 0.165 \leq y \leq 0.311 \\ -8.727 - 100.994e^{-0.760(y-m)} \\ +110.515e^{-0.694(y-m)} & 0.311 \leq y \leq 0.698 \\ \vdots \\ 0.001 - 93.350e^{-1.749(y-m)} \\ +94.235e^{-1.744(y-m)} & 1.822 \leq y \leq 6.050 . \end{cases}$$

The normalization constant k_Y is 0.971844, so the normalized MTE distribution is $\hat{f}_Y(y) = \hat{f}_Y^{(u)}(y)/0.971844$. The MTE approximation to the $LN(2.7, 0.6^2)$ PDF for X is

$$\hat{f}_X(x) = (1/14.8797) \cdot \hat{f}_Y((1/14.8797) \cdot x) .$$

The result is shown in Figure 2 overlaid on the actual $LN(2.7, 0.6^2)$ PDF along with two other distributions used in Section 4 (it is the distribution with the second highest mode).

4 Inventory Management Examples

In this section we consider the problem of establishing a (Q, R) policy in a continuous review inventory system. In this model, Q is the order quantity and R is the inventory level when an order is placed, or *reorder point*. The expected cost function (Hadley and Whitin, 1963) is

Table 1: Calculation of parameters for an MTE distribution approximating the $LN(0, 0.6^2)$ PDF.

k	5	4	3	2	1	
σ^{k-1}	0.1296	0.216	0.36	0.6	1.0000	\hat{a}_{j1}
$\hat{c}_{k11}^{(3)}$	-12.8729	19.2704	-10.3045	2.2788	-0.2493	-0.098
$\hat{c}_{k21}^{(3)}$	218.6796	-378.9597	241.8550	-68.0000	95.0700	87.823
$\hat{c}_{k31}^{(3)}$	606.9384	-1120.0537	785.4244	-252.0248	41.9427	10.208
$\hat{c}_{k41}^{(3)}$	185.4334	-322.1284	206.3402	-58.3289	-107.4016	-113.664
$\hat{c}_{k51}^{(3)}$	852.5394	-1556.6422	1075.2962	-338.6557	53.4414	11.609

$$TC(q, r) = \frac{KY}{q} + \frac{\pi Y S_R(r)}{q} + c \left(\frac{q}{2} + r - E(X) \right). \quad (7)$$

In this formula, K is the fixed cost per order, Y is the annual demand, c is the unit holding cost per year, and π is the stockout cost per unit. The terms are the annual ordering, stockout, and holding costs, respectively. The third term assumes the firm holds an average *safety stock* of $R - E(X)$. The shape of the distribution for X enters the model via the expected shortage per cycle $S_R(r)$. This is approximated as

$$\hat{S}_R(r) = \int_r^\infty (x - r) \cdot \hat{f}_X(x) dx \quad (8)$$

where $\hat{f}_X(x)$ is an approximation to the lead time demand (LTD) distribution.

4.1 Uncertain Demand

Das (1983) presents an approximate method for finding a (Q, R) policy where LTD is log-normal. Presumably, an approximately log-normal PDF for LTD could be the result of demand per unit time for a fixed lead time being log-normal so that LTD is a sum of log-normals that can be approximated using the FW method. In other words, Das presents results that can be used to find an approximately optimal Q and R after σ_X and μ_X from (3) and (4) have been determined. The test cases in Table 2 are used by Das, with $K = 30$ in each case, $\pi = 5$ for Cases 1 and 2, and $\pi = 6$ for Case 3.

Das specifies the values in the μ_X and σ_X columns of Table 2. The values in the μ_{X_ℓ} and σ_{X_ℓ} columns are calculated assuming that lead time is fixed at $L = 5$ and that LTD is the i.i.d. sum of $L = 5$ one period demand values that are log-normally distributed as $LN(\mu_{X_\ell}, \sigma_{X_\ell}^2)$. The

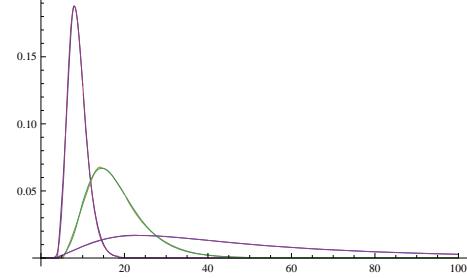


Figure 2: MTE approximations and actual log-normal PDFs for the Das (1983) test cases.

Q_D and R_D columns contain the approximately optimal solutions obtained by Das.

To use the MTE approximation to find optimal Q and R for the three test cases, we approximate the three log-normal PDFs with μ_X and σ_X shown above using the process from Section 3.4. These three MTE distributions are shown in Figure 2, overlaid on the actual log-normal PDFs. The function with the largest mode corresponds to $\mu_X = 1.6$ and $\sigma_X = 0.8$, whereas the function skewed the farthest right approximates the $LN(2.3, 1)$ PDF.

Using the MTE approximations as the $\hat{f}_X(x)$ PDFs for LTD allows us to establish a closed-form expression for $\hat{S}_R(r)$ in (8). For example, in Case 1, the expression is

$$\hat{S}_R(r) = \begin{cases} 78381.9 - 429.776r + 0.809782r^2 \\ + 75472.7e^{-0.0101464r} - 153835e^{-0.00776065r} & 17.2631 \leq r < 27.1126 \\ 0.228577 - 0.00531733r + 0.0000306374r^2 \\ - 1582.93e^{-0.117546r} + 1601.78e^{-0.117187r} & 27.1126 \leq r \leq 90.0171 . \end{cases} \quad (9)$$

Table 2: Parameters and solutions for the Das (1983) test cases.

Case	Y	μ_X	σ_X	μ_{X_ℓ}	σ_{X_ℓ}	h	(Q^*, R^*)	(Q_D, R_D)	(Q_M, R_M)	HD_D	HD_M	CT_M
1	400	2.7	0.6	0.69	1.07	4	(90.0, 24.8)	(89.7, 25.8)	(88.6, 25.2)	0.03%	0.05%	1.4s
2	100	1.6	0.8	-0.54	1.30	2	(62.0, 8.1)	(61.9, 9.5)	(60.7, 8.6)	0.24%	0.00%	1.5s
3	300	2.3	1	0.06	1.50	3	(105.4, 23.7)	(104.7, 25.5)	(95.8, 26.0)	0.05%	0.18%	1.3s

Once the $\hat{S}_R(r)$ expression is established we use it to approximate $TC(q, r)$ in (7), and we can subsequently find the values for Q and R that minimize $\hat{TC}(q, r)$. We perform this optimization in Mathematica 8.0. All computations were done on a PC with 16GB of memory and an Intel Core2 Duo 2.93 GHz processor.

For comparison purposes, a simulation-optimization technique is employed to find the “exact” solution. Using Oracle Crystal Ball software’s tabu-search driven OptQuest tool, this approach tests numerous possible values for Q and R . Demand is simulated for each day, then total cost is calculated by computing shortage cost as $\text{Max}\{0, R - X\} \cdot (\pi Y/Q)$ on each simulation trial. To find an accurate solution, we used 10,000 trials to test each selected (Q, R) combination and ran OptQuest for several hours. For Case 1, the exact solutions are determined to be $Q^* = 90$, $R^* = 24.8$. The expected total cost using a longer simulation of 1,000,000 trials with the optimal values is 390.7.

To judge the effectiveness of the MTE approach versus the exact solutions, the cost differential used by Heuts et al. (1986) can be calculated as

$$HD = 100\% \cdot \frac{TC(Q_M, R_M) - TC(Q^*, R^*)}{TC(Q^*, R^*)}. \quad (10)$$

This expression uses the MTE solutions, Q_M and R_M , and simulates the expected total cost calculated over 1,000,000 trials. Thus, HD measures the error that occurs when the MTE solutions are employed as an approximation to the true solutions. The HD cost differential is calculated similarly for Q_D and R_D .

Solutions found in all three test cases defined by Das are shown in Table 2 (M and D subscripts correspond to the MTE and Das methods, respectively). Computing time for the MTE method, CT_M , is also reported. Das does not report computing time required to ob-

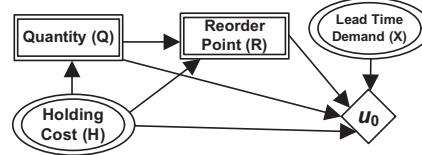


Figure 3: Influence diagram model.

tain solutions using the analytical method, but does state that they are about 20% of that required for an iterative method due to Hadley and Whitin (1963). Actual computational times for the Das method are likely negligible.

The Das solution has a lower HD in two cases, whereas the MTE solution is more accurate in the second case. In summary, the MTE method performs comparably with the Das method and the simulation-optimization method for these problems and uses minimal computation time.

4.2 Uncertain Demand and Holding Cost

Another motivation for approximating the LTD distribution with an MTE potential is to use IDs designed for decision-making with continuous variables. In such models, the functional form of the PDFs and utility functions must be such that the mathematical operations of addition, multiplication, and integration maintain a closed-form result that can be used in further calculations.

Suppose that we have a problem similar to the Das Case 1 example, but that holding cost is a RV because the company finances inventory purchases with a variable interest rate line of credit. Before each order, the company knows the true current holding cost and can adjust its order accordingly. The ID model for this problem appears in Figure 3. Chance, decision, and utility nodes appear as ovals, rectangles, and diamonds, respectively.

The utility function in the problem is

$$u_0(q, r, h) = -\frac{K}{Y} \cdot g_1(q) - \pi \cdot Y \cdot \hat{S}_R(r) \cdot g_1(q) - g_2(h) \cdot (0.5 \cdot g_3(q) + g_4(r) - E(X)) , \quad (11)$$

where $g_1(q) = 1/q$, $g_2(h) = h$, $g_3(q) = q$, and $g_4(r) = r$. This is similar to $TC(q, r)$ in (7), except that negative signs have been placed in front of each term and holding cost is now a RV. The utility function is not explicitly a function of X because the term $S_R(r)$ is created by integrating out X . We include X in the graphical ID to emphasize that the PDF for X affects the total cost through $\hat{S}_R(r)$. The computation that creates the function $\hat{S}_R(r)$ becomes a marginalization operation for X .

The LTD distribution is the same MTE distribution used in Section 4.1 for Case 1. Holding cost, H , is $N(4, 4/9)$, and is approximated with an MTE distribution (Cobb et al., 2006) on the interval $[2, 6]$. u_0 is replaced with an MTE utility function, u_1 , by approximating each function g_1, \dots, g_4 with an MTE potential, then combining the result. For example, $g_1(q) = 1/q$ is approximated with an MTE potential $\hat{g}_1(q) = 0.0059 + 0.0452 \cdot \exp(-0.024q)$ by finding constants that minimize the mean squared error (MSE) between g_1 and \hat{g}_1 as follows:

$$\underset{a_1, a_2, a_3}{\text{ArgMin}} \int_{q_{min}}^{q_{max}} \left(\frac{1}{q} - a_1 - a_2 \exp(a_3 q) \right)^2 .$$

The values $q_{min} = 63$ and $q_{max} = 110$ are logical lower and upper bounds for Q because values minimize total cost at expected demand at $H = 2$ and $H = 6$. The other functions are similarly approximated.

The expression in (9) for $\hat{S}_R(r)$ can be easily approximated by an MTE function. We find $g_5(r) = -0.0011 + 20.8 \exp(-0.0961r)$ as the function that minimizes the MSE between $\hat{S}_R(r)$ and $g_5(r)$. When the results of the constants in the problem and the approximations to g_1, \dots, g_4 and $\hat{S}_R(r)$ are combined, the result is an MTE function (Moral et al., 2001).

To solve the ID, the variables are deleted in the sequence X, R, Q, H . This example uses the algorithm (Cobb, 2010) that facilitates the determination of a piecewise-linear decision rule

for a continuous decision variable with multiple continuous parents. In this case, removing R requires that we determine a decision rule for $R = f(q, h)$. The actual values of Q and H will be known when choosing R .

This decision rule is

$$\hat{\theta}_1(q, h) = \begin{cases} -4.658h - 0.135q + 56.995 & 2.0 \leq h < 2.5 \\ -4.658h - 0.129q + 56.462 & 2.5 \leq h < 3.0 \\ \vdots \\ -3.593h - 0.097q + 47.919 & 5.5 \leq h \leq 6. \end{cases}$$

As Q increases given an established H , the optimal r should be set to a lower value. We divide Ω_H into 8 regions to allow the slope of the piecewise linear decision rule to change for a given value of Q as H increases. We could use more regions to improve accuracy, or fewer regions to speed up the solution process. To complete the elimination of R , we calculate $u_2(q, h) = u_1(q, \hat{\theta}_1(q, h), h)$. The result is an MTE potential because we are substituting a linear function for R into the MTE potential u_1 (Cobb, 2010).

Applying the decision variable marginalization procedure when a variable has only one continuous parent, as in the case for Q , results in a piecewise linear decision rule as follows: $\hat{\theta}_2(h) = 131.631 - 10.6818h$ for $2 \leq h < 3.4$, $\hat{\theta}_2(h) = 126.043 - 9.03846h$ for $3.4 \leq h < 4.05$, and $\hat{\theta}_2(h) = 119.18 - 7.34375h$ for $4.05 \leq h \leq 6$.

Suppose we find before placing an order that $H = 4$. Since $\hat{\theta}_2(4) = 89.9$, we should order $Q = 89.9$ units. Since $\hat{\theta}_1(89.9, 4) = 26.3$, we reorder with $R = 26.3$ units remaining. The expected total cost is 387.0.

5 Discussion

This paper has established a process for estimating the distribution of a sum of i.i.d. log-normal RVs (a single log-normal RV is a special case). The procedure uses the Fenton-Wilkinson approximation (Fenton, 1960) to estimate the parameters for a single log-normal PDF that approximates the sum of log-normal RVs. Once these parameters are determined, an MTE distribution is determined to approximate the PDF with $\mu = 0$ and the FW σ^2 parameter. The method of convolutions is used to find

an MTE distribution that approximates that of the sum of the log-normal RVs.

To reduce the tabular inputs required to implement the MTE approximation, the parameters (constants) required to approximate log-normal PDFs with $\mu = 0$ and $0.05 \leq \sigma \leq 1$ are stated as polynomial functions of σ . This covers a range of likely values for the inventory applications presented in the paper, and the FW approximation has been deemed to be effective for these values of σ .

The MTE approach to approximating the distribution for LTD in an inventory control problem was compared to an existing approach to the literature (Das, 1983). One advantage of the MTE method is that a closed-form is obtained for the LTD distribution and the expected shortage per cycle. This advantage was demonstrated when the MTE approximation was subsequently used in an ID model to extend the analysis of the inventory problem to a situation where holding cost is a RV.

Possibilities for future research are as follows. First, the MTE approximation can be extended to cases with larger σ^2 parameters to model applications in such areas as telecommunications and risk analysis. Second, the FW method extends to the case where the component log-normal PDFs are not i.i.d., so the MTE approximation for a sum of log-normals could be utilized in applications with sums of log-normal PDFs that are not identical. Third, other approximations, such as the mixtures of polynomials approach (Li and Shenoy, 2012), could be compared with the MTE approach. Third, extensions of the ID approach to more complicated inventory problems can be pursued.

Acknowledgments

This research has been partially funded by the Spanish Ministry of Economy and Competitiveness, through project TIN2010-20900-C04-02, by Junta de Andalucía through project TIC-7821 and by ERDF funds.

References

N.C. Beaulieu, A.A. Abu-Dayya, and P.J. McLane. 1995. Estimating the distribution of a sum of independent log-normal random variables. *IEEE*

- Transactions on Communications*, 73(12):2869–2873.
- B. R. Cobb, P. P. Shenoy, and R. Rumí. 2006. Approximating probability density functions in hybrid Bayesian networks with mixtures of truncated exponentials. *Statistics and Computing*, 16(3):293–308.
 - B. R. Cobb. 2010. Continuous decision variables with multiple continuous parents. In P. Myllymäki, T. Roos, and T. Jaakola, editors, *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*, pages 97–104, Helsinki, Finland.
 - C. Das. 1983. Inventory control for log-normal demand. *Computers and Operations Research*, 10(3):619–629.
 - L. F. Fenton. 1960. The sum of log-normal probability distributions in scatter transmission systems. *IRE Transactions on Communication Systems*, 8:57–67.
 - G. Hadley and T. Whitin. 1963. *Analysis of Inventory Systems*. Prentice-Hall, Englewood Cliffs, N.J.
 - R. Heuts, K. van Lieshout, and K. Baken. 1986. An inventory model: What is the influence of the shape of the lead time demand distribution? *Mathematical Methods in Operations Research*, 30(2):B1–B14.
 - H. Langseth, T.D. Nielsen, R. Rumí, and A. Salmerón. 2012. Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53(2):212–227.
 - Y. Li and P. P. Shenoy. 2012. A framework for solving hybrid influence diagrams containing deterministic conditional distributions. *Decision Analysis*, 9(1):55–75, March.
 - S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In S. Benferhat and P. Besnard, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Lecture Notes in Artificial Intelligence 2143, pages 156–167. Springer, Berlin.
 - V. Romero, R. Rumí, and A. A. Salmerón. 2006. Learning hybrid bayesian networks using mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 42(1-2):54–68.
 - S.C. Schwartz and Y.S. Yeh. 1982. On the distribution function and moments of power sums with log-normal components. *Bell System Technical Journal*, 61(7):1441–1462.

A Bayesian Network model for predicting the outcome of in vitro fertilization

Giorgio Corani

IDSIA - Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (Manno, Switzerland)
giorgio@idsia.ch

Cristina Magli

IIRM - International Institute for Reproductive Medicine (Lugano, Switzerland)

Alessandro Giusti

IDSIA - Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (Manno, Switzerland)

Luca Gianaroli

IIRM - International Institute for Reproductive Medicine (Lugano, Switzerland)

Luca Gambardella

IDSIA - Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (Manno, Switzerland)

Abstract

We present a Bayesian network model for predicting the outcome of in-vitro fertilization (IVF). The problem is characterized by a peculiar missingness process, and we propose a simple but effective averaging approach which improves parameter estimates compared to the traditional MAP estimation. The model can provide relevant insights to IVF experts.

1 Introduction

According to the World Health Organization, infertility affects more than 80 million people worldwide; in vitro fertilization (IVF) helps addressing the problem. In IVF, a semen specimen is merged with a female egg in laboratory to yield, after some days of culture, an *embryo*. Several embryos are cultured for each woman: after some days, some or all of them are transferred to the woman. A clinical pregnancy occurs when at least one of the transferred embryos implants. Predicting the outcome of an IVF transfer is a challenging problem, in which models generally achieve only limited accuracy (Saith et al., 1998; Morales et al., 2008).

A pioneering approach for estimating the probability of single pregnancy and multiple pregnancy after an IVF transfer is the EU model (Speirs et al., 1983), which assumes that, for pregnancy to happen, both a *receptive* uterus and a *viable* embryo are necessary. We repre-

sent *uterine receptivity* as the binary variable U , with states $\{u, \neg u\}$ (u denoting receptivity, $\neg u$ non-receptivity); we represent *embryo viability* as the binary variable E , with states $\{e, \neg e\}$ (e denoting viability, $\neg e$ non-viability). We denote by θ_e and θ_u respectively the probabilities of the embryo to be viable, namely $\theta_e = P(E = e)$, and $\theta_u = P(U = u)$. The EU model estimates the probability of pregnancy after the transfer of a *single* embryo as $\theta_e \cdot \theta_u$, thus assuming the independence of viability and receptivity. When dealing with the transfer of *multiple* embryos, each embryo is assumed to implant independently from the others. For instance, if *two* embryos are transferred, the probability of a single pregnancy is $2\theta_u\theta_e(1 - \theta_e)$, accounting for the fact that two embryos can give rise to pregnancy; the probability of double pregnancy is instead $\theta_e^2\theta_u$. The *EU assumption* is thus that the number of babies born after an IVF transfer is $(U = u) \cdot \sum(E_i = e)$, where E_i

is the viability of the i -th transferred embryo. The main limitation of the original EU model is the unrealistic assumption of θ_e and θ_u being identical for respectively all women and all embryos. Therefore, in (Zhou and Weinberg, 1998) the model has been reworked (adopting a generalized linear model framework) by letting vary both θ_u and θ_e on external covariates; in particular, by letting θ_u depend on the age of the woman and θ_e on the number of cells which the embryo contains (the number of cells contained by an embryo is considered as a marker of its implantation capability). More recently it has been investigated (Roberts et al., 2010) how to select the number and the types of covariates on which θ_u and θ_e should depend. In fact, quantifying how θ_e and θ_u vary with the external covariates such as the age of the woman or the embryo score is an important by-product of the models based on the EU assumption, which allows for instance verifying the effectiveness of the adopted embryo scoring systems, a very important problem for embryologists.

Analyzing the IVF data under the EU assumption implies having to deal with a *partial observability* problem. For instance, if pregnancy does *not* occur, it cannot be ascertained whether a) the uterus was *non-receptive*, b) *all* the transferred embryos were *non-viable* or c) both. If pregnancy occurs, the embryo is known to be receptive, but it is still unknown which of the embryos gave rise to the pregnancy, unless the number of babies equals the number of transferred embryos. The partial observability problem is addressed by (Zhou and Weinberg, 1998) adopting a latent variable formulation, and then estimating the parameters of the model via Expectation-Maximization (EM).

The contributions of this paper are as follows: a) a novel probabilistic model of IVF transfers, based on a Bayesian network; b) a simple but effective averaging approach to improve the estimate of the parameters from the incomplete samples which characterize the problem; c) a thorough experimental analysis showing good predictive performance of the proposed model on both artificial and real data sets.

2 The model

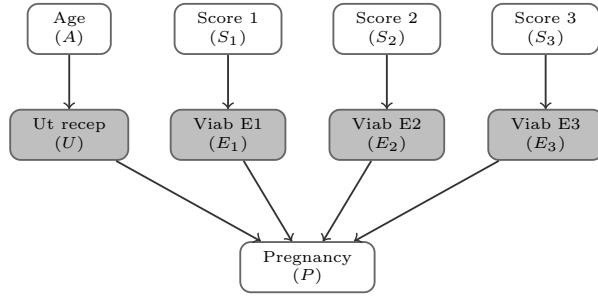


Figure 1: The IVF model: nodes affected by the missingness process are shown with a gray background.

Given a generic variable X , we denote by θ_X the probability mass function which associates a marginal probability to each different value of X ; we denote by $\theta_{X|Pa(X)}$ the probability mass function which associates a conditional probability to each different value of X , given each possible configuration of the parents of X , denoted as $Pa(X)$. We moreover denote by $\boldsymbol{\theta}$ the set of all the parameters of the BN model.

We represent the IVF transfer by the BN model shown in Fig. 1. The model manages IVF cycles with up to three embryos, as this is the maximum allowed under the Swiss law; however, it can be straightforwardly extended to manage a higher number of transferred embryos. The woman age is discretized as $\{<34, 34-40, 40+\}$.

We denote by \mathcal{S} the set of nodes $\{S_1, S_2, S_3\}$ which represent the score of the embryos and which are referred to as the \mathcal{S} -nodes. The score can be *top* or *non-top* depending on the morphology of the embryo, according to criteria which are out of our scope. Moreover, embryos graded as *top* consistently on all the days of the culture are scored as *top-history*. The score of the embryos is also allowed to be *no-transfer*, thus allowing to model cycles with less than 3 transferred embryo: since multiple pregnancies are dangerous for the health of both mother and babies, often only 1 or 2 embryos are transferred. Summing up, the \mathcal{S} -nodes take values in $\{\text{no-transfer}, \text{non-top}, \text{top}, \text{top-history}\}$. The embryos occupy the different positions (1,2,3)

in a purely random fashion.

The S -nodes are *tied*: they share the same mass function θ_S instead of having separate mass functions θ_{S_1} , θ_{S_2} and θ_{S_3} . This prevents the same embryo score (e.g., top) being given a different marginal probability depending on whether one refers to node S_1 , S_2 or S_3 . Moreover, sharing the parameters allows to obtain more accurate estimates than using a different mass function for each S -node. Node U represents uterine receptivity; it is therefore binary, with states $(u, \neg u)$. We denote by \mathcal{E} the set of nodes $\{E_1, E_2, E_3\}$, which are referred to in the following as \mathcal{E} -nodes. Each \mathcal{E} -node represents the viability of a different embryo; it is thus binary with states $(e, \neg e)$. The \mathcal{E} -nodes share the parameter set of the conditional mass function $\theta_{E|S}$, rather than having independent mass functions $\theta_{E|S_1}$, $\theta_{E|S_2}$ and $\theta_{E|S_3}$. Again, this prevents two embryos with the same score being given different probability of being viable just because they occupy a (random) different position; moreover, it enables more accurate estimate than using a different set of parameters for each \mathcal{E} -node.

The pregnancy node P has four states $\{0, 1, 2, 3\}$, corresponding to the number of babies which might be born after having transferred up to three embryos. The CPT of P encodes the deterministic EU assumption, namely it assigns probability 1 to the state whose value equals $(U = u) * \sum_{i=1}^{i=3} (E_i = e)$.

In the following, we discuss the incompleteness which characterizes the instances. Node P represents the class; it is thus always observed in training and always missing in test. The missingness process affecting the U and the \mathcal{E} -nodes is instead more complicated. Let us consider an IVF cycle in which all the 3 embryos are transferred. Let us start by the *training stage*; recall that at training stage P is always observed. Given $P = 0$, the observation of node U is missing; given $P > 0$, U is observed ($U = u$). Given $P < 3$, the observation of the \mathcal{E} -nodes is missing; given $P = 3$, the \mathcal{E} -nodes are observed ($E_i = e \forall i$). Thus, the observation of U and the \mathcal{E} -nodes is missing or not depending on the value of the observed variable P ; the missing-

ness process is MAR (missing at random). Since most IVF cycles result in no-pregnancy, in most instances both U and the \mathcal{E} -nodes are *not* observed.

At *test* stage, the parameters have been already learned and the goal is to assess the predictive ability of the model. The U and the S -nodes are *never* observed at test stage (if they were observed at test stage, the outcome would be known with certainty); they are therefore affected by a MCAR (missing completely at random) missingness process. For more details about MAR and MCAR missing data, see (Koller and Friedman, 2009, Chap.19).

In some cycles less than three embryos are transferred, to reduce the danger of multiple pregnancy. For these cycles, the missingness process affects the \mathcal{E}_t -nodes rather than the \mathcal{E} -nodes, the \mathcal{E}_t -nodes representing the viability of the *transferred* embryos. The \mathcal{E}_t -nodes are affected by a MAR and a MCAR missingness process at respectively train and test stage. The viability of non-transferred embryos is alway observed (as $\neg e$), since a non-transferred embryo is by definition non-viable.

3 Estimation procedure

Given a generic variable X , we denote by θ_X^x the probability $P(X = x)$ and by $\theta_{X|Y}^{x|y}$ the probability $P(X = x|Y = y)$; this additional notation allows accessing singletons of the mass functions. We denote as \mathcal{X} the set of all variables which constitute the BN model and by \boldsymbol{x} an *instance*, namely a single row of data containing either an observation or a missing value for each variable.

Because of the incomplete samples, the likelihood contains the summation over all the possible data completions. As an example, consider the following instance \boldsymbol{x} of the training set:

A	U	S_1	S_2	S_3	E_1	E_2	E_3	P
40+	u	top	ntop	toph	?	?	?	1

in which a single pregnancy has occurred; thus, the uterus is known to be receptive ($U=u$) but the observations of the \mathcal{E} -nodes is missing,

since it is unknown which of the three embryos has implanted. The possible data completions are those in which exactly one out of three embryos is viable; the likelihood of the instance is thus:

$$\begin{aligned} P(\mathbf{x}|\boldsymbol{\theta}) = & \theta_A^{40+} \cdot \theta_U^u \cdot \theta_S^{top} \cdot \theta_S^{ntop} \cdot \theta_S^{troph} \cdot \\ & \cdot [\theta_{P|U,\mathcal{E}}^{1|u,e_1,\neg e_2,\neg e_3} \cdot \theta_{E|S}^{e|top} \cdot \theta_{E|S}^{\neg e|ntop} \cdot \theta_{E|S}^{\neg e|troph} \\ & + \theta_{P|U,\mathcal{E}}^{1|u,\neg e_1,e_2,\neg e_3} \cdot \theta_{E|S}^{\neg e|top} \cdot \theta_{E|S}^{e|ntop} \cdot \theta_{E|S}^{\neg e|troph} \\ & + \theta_{P|U,\mathcal{E}}^{1|u,\neg e_1,\neg e_2,e_3} \cdot \theta_{E|S}^{\neg e|top} \cdot \theta_{E|S}^{\neg e|ntop} \cdot \theta_{E|S}^{e|troph}] \end{aligned}$$

Notice that the likelihood contains terms θ_S^{top} , θ_S^{ntop} and θ_S^{troph} rather than $\theta_{S_1}^{top}$, $\theta_{S_2}^{ntop}$ and $\theta_{S_3}^{troph}$, as a consequence of the \mathcal{S} -nodes sharing the same mass function. The same consideration applies to the \mathcal{E} -nodes; in the likelihood it appears e.g. $\theta_{E|S}^{\neg e|top}$ rather than $\theta_{E_1|S}^{\neg e|top}$.

The log-likelihood for the whole training set is obtained by summing the logs of the likelihood of all instances; it has a complex expression, which would be very difficult to analytically optimize. However, recalling that the missingness process at the training stage is MAR, the Expectation-Maximization algorithm (EM) can be used to learn the parameters. We estimate the parameters by maximizing the posterior probability of the data $P(\boldsymbol{\theta}|D)$ rather than the likelihood; this approach improves the estimates and reduces the danger of overfitting. In particular, we adopt a Dirichlet prior for the parameters, setting to 1 the equivalent sample size. In the following, we refer to $P(\boldsymbol{\theta}|D)$ as the *MAP score*.

Given the multi-modality of $P(\boldsymbol{\theta}|D)$, EM converges only to a *local* maximum of the MAP score. It is thus common to initialize EM from m different starting points and to execute m different *EM runs*, to eventually select the estimate yielding the highest MAP score. In the following, we refer this procedure as *MAP estimation*. By definition, MAP estimation selects the parameterization which is the most probable a posteriori, rather than integrating over the full posterior: for this reason, it “*does not offer the same benefits as a full Bayesian estimation. It does not attempt to represent the shape*

of the posterior and thus does not differentiate between a flat posterior and a sharply peaked one.” (Koller and Friedman, 2009, Sec.17.4.4). In particular, MAP estimation is a good approximation of Bayesian estimation when the posterior is sharply peaked around the maximum; this is however not the case when learning from incomplete samples. Typically, different EM runs achieve close values of the MAP score, returning however very different parameter estimates. Therefore, MAP estimation in this context is hardly robust. This consideration remain valid even if informative starting point are adopted for EM, which allows improving the estimates.

As an alternative to MAP estimation, we propose the following averaging approach. With reference to a generic parameter θ_X^x , we average its estimates obtained in the m EM runs as follows:

$$\hat{\theta}_X^x = \frac{\sum_{i=1}^{i=m} \hat{\theta}_X^{x-i} P(\hat{\boldsymbol{\theta}}^i|D)}{\sum_{i=1}^{i=m} P(\hat{\boldsymbol{\theta}}^i|D)} \quad (1)$$

where $\hat{\theta}_X^{x-i}$ and $P(\hat{\boldsymbol{\theta}}^i|D)$ denote respectively the estimate of θ_X^x and the MAP score obtained in the i -th EM run, once it has converged. We average all parameters of the model according to Equation (1).

To illustrate the rationale of our approach, consider the general query $P(\mathcal{Z}|\mathbf{y}, D)$, where \mathcal{Z} is the set of variables being queried, and \mathbf{y} is the evidence available on the subset of variables $\mathcal{Y} \in \mathcal{X}$. A fully Bayesian inference would be:

$$P(\mathcal{Z}|\mathbf{y}, D) = \int P(\mathcal{Z}|\mathbf{y}, D, \boldsymbol{\theta}) P(\boldsymbol{\theta}|D) d\boldsymbol{\theta} \quad (2)$$

while, under MAP estimation, the above integral is roughly approximated as:

$$P(\mathcal{Z}|\mathbf{y}, D) \approx P(\mathcal{Z}|\mathbf{y}, D, \hat{\boldsymbol{\theta}}) \quad (3)$$

where $\hat{\boldsymbol{\theta}}$ represents the most probable parameters estimate a posteriori.

The following *pseudo-Bayesian* approach moves towards the Bayesian inference, by sam-

pling the posterior in correspondence of the local maxima identified by the m EM runs:

$$P(\mathcal{Z}|\mathbf{y}, D) \simeq \sum_{i=1}^{i=m} P(\mathcal{Z}|\mathbf{y}, D, \hat{\boldsymbol{\theta}}^i) P(\hat{\boldsymbol{\theta}}^i|D) \quad (4)$$

The pseudo-Bayesian approach should generate more accurate inferences than the MAP approach, because it partially reconstructs the shape of the posterior. A similar idea has been for instance used with good results in clustering with naive Bayes (Santafé et al., 2006). Yet, it requires keeping a collection of e.g. $m=20$ networks, each characterized by the same structure but different parameters; this compromises the possibility for IVF experts of easily interpreting the model.

The averaging idea of Equation (1) aims at keeping as much as possible the benefits of the pseudo-Bayesian approach, but instantiating only a single network. In particular, averaging the parameters according to Eq. (1) produces the same inferences of the pseudo-Bayesian approach of Eq. (4), in case of a query of type $P(X = x|pa(X))$, where $pa(X)$ denotes an instantiation of all the parents of X . In these cases, the returned inference correspond in fact to the parameter $\theta_{X|Pa(X)}^{x|pa(X)}$ of the network; averaging the parameters according to Eq. (1) is equivalent to averaging the inferences according to Eq. (4). This is especially important for our application, in which IVF experts are interested in analyzing the estimates of the conditional mass functions $\theta_{E|S}$ and $\theta_{U|A}$.

However, a single network with parameters averaged according to Eq.(1) does not yield the same inferences than the pseudo-Bayesian approach of Eq.(4) in more general queries. In general, it is not possible replicating by a single network the inference produced by a set of networks. Moreover, the property of parameter decomposability, which allows estimating independently the different conditional probability mass function, does not hold if the training set is incomplete (Koller and Friedman, 2009, Chap. 19.1.3). This prevents in principle averaging the parameters referring to the same conditional mass functions across the different EM

runs.

Yet the experiments of the next section show that the averaging approach consistently outperforms MAP estimation, both as for the accuracy of the parameters estimates and of the predictive inferences: the benefits of going towards Bayesian estimation outweigh the effects of the introduced approximations.

4 Experiments with generated data

We perform experiments with generated data, in order to compare the estimates generated by MAP estimation and by the averaging approach. We consider the network structure of Figure 1 and the sample sizes $n \in \{50, 150, 300, 450, 600\}$. For each sample size n , we perform 100 *experiments* constituted by the following steps: a) random drawing of the parameters of the structure, thus instantiating the *true network*; b) sampling of n complete instances (*training set*) from the true network; c) application of the MAR missingness process of the training stage, described in Section 2; d) execution of EM from $m=20$ different initializations and estimation of the parameters adopting the MAP and the averaging approach; e) evaluation of the estimated parameters; f) generation of a test set of 1000 instances, making missing the U and the \mathcal{E} -nodes; g) classification of the test instances. On average, in the simulations U is missing in 75% of the instances and the \mathcal{E} -nodes in 80% of the instances.

Since the problem has 4 classes, we measure the classification performance by computing 4 AUCs: one for each of no-pregnancy, single, double and triple pregnancy; they are denoted as $AUC_0, AUC_1, AUC_2, AUC_3$. For this problem, AUC is a more appropriate measure than accuracy: typically, some 70-80% of the cycles ends with non-pregnancy and thus a *trivial* predictor which always returns no-pregnancy would achieve an *apparently* high accuracy of 70-80%. Previous studies in this area (Saith et al., 1998; Morales et al., 2008) show that even sophisticated classifiers only achieve a small improvement of accuracy over the trivial predictor, whose performance is inflated by the skew in the

class distribution. The AUC is insensitive on the class distribution and thus allows assessing more clearly the performance of the classifiers.

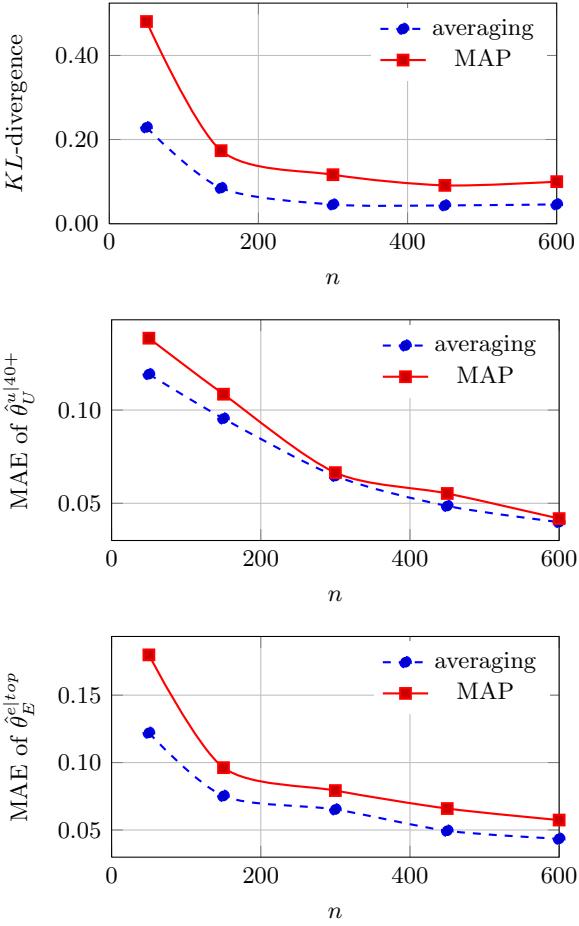


Figure 2: Experimental results on simulated data; the averaging approach *reduces* the KL-divergence from the true model and thus the mean absolute error (MAE) in the estimation of the parameters. Each point represents the average over 100 experiments.

The averaging approach largely reduces, compared to the MAP estimation, both the mean and the standard deviation of the KL-divergence from the true network, as shown in Figure 2. The reduction of the KL-divergence is significant at *each* sample size (*t*-test, $p < 0.01$). For instance the mean KL-divergences are for $n=50$: 0.22 ± 0.12 for the averaging approach and 0.48 ± 0.35 for MAP estimation; for $n=600$: 0.03 ± 0.05 and 0.05 ± 0.09 . Thus even for large

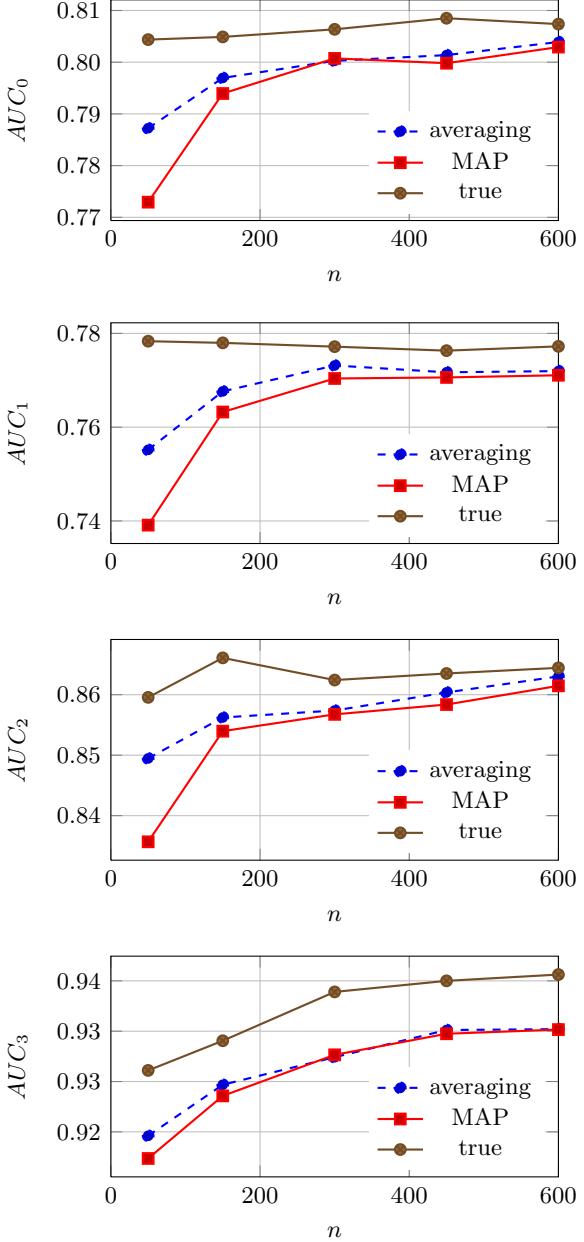


Figure 3: Experimental results on simulated data; the averaging approach *increases* the AUCs compared to MAP estimation. Each point represents the average over 100 experiments.

n , the averaging approach reduces of about 40% the mean KL-divergence, although this is hard to grasp from Figure 2, because of its scale. The reduction in the KL-divergence implies a much better estimate of the conditional mass func-

tions $\theta_{E|S}$ and $\theta_{U|A}$. The lower plots of Figure 2 show the reduction of the mean absolute error in the estimation of two (randomly chosen) parameters taken from $\theta_{E|S}$ and $\theta_{U|A}$.

The improved estimates result in better predictive inferences about the probability of pregnancy, as measured on the test sets. The averaging approach improves all AUCs over the MAP estimation, as shown in Figure 3; the improvement is however smaller in percentage than on the parameter estimates. Considering 4 different AUCs and 5 sample sizes, there are 20 possible combinations n -AUC; in 7 out of such 20 cases the AUC improvement yielded by the averaging approach is significant (t -test, $p < 0.01$).

Another interesting finding is that the AUC of the *true* model is generally not very far from that of the *estimated* models: the average AUC (averaging $AUC_0, AUC_1, AUC_2, AUC_3$ over all experiments) is 83.4, 83.7 and 84.5 for respectively the MAP, the averaging approach and true model. The point is that at test stage, as already discussed, U and the \mathcal{E} -nodes are never observed; this is the main difficulty of predicting the outcome of IVF cycles, under the EU assumption. There is thus a major limit on predictive performance, which holds also if the model parameters are perfectly known.

4.1 Analysis of a real data set

We analyze 388 cycles performed at the International Institute for Reproductive Medicine (IIRM) of Lugano. The percentage of no pregnancy, single pregnancy and twin pregnancy is respectively 80%, 16% and 4%; no triple pregnancies are present.

It is not possible measuring the KL-divergence of the estimated networks from the true model, which is indeed unknown. However, we exploit cross-validation to provide some results about the quality of the parameter estimates. By adopting a 5-folds cross-validation, 4/5 of the instances (310) are used to learn the model, and the remaining ones to test the predictions. We consider as a *reference model* the BN model learned on the full data set of 388 instances. We repeat 10 times the cross-validation; this yields a sample of 50 (5 folds \cdot 10

repetitions) KL-divergences between the models estimated on the sampled training sets and the reference model. We stratify training and test sets, which contain the same proportion of no-pregnancy, single pregnancies and twin pregnancy of the complete data set. The averaging approach reduces the mean KL-divergence from the reference model of about 5% compared to MAP estimation; this reduction is significant (t -test, $p < 0.01$); moreover, it also reduces the standard deviation of about 15%. The advantage is larger if a smaller training set is available; repeating the same procedure with a 2-folds cross-validation (194 instances in the sampled training set), the averaging approach reduces the mean KL-divergence of about 42% and the standard deviation of about 75%. To avoid any bias in favor of the averaging approach, the reference model has been estimated using the MAP approach.

We report in the *first* column of Table 1 the AUCs measured by 5-folds cross-validation, adopting the averaging approach (the model is indicated as BN-EU); they are *not* significantly different from those obtained using MAP estimation. In fact, the training set is quite large, and estimation methods tend to converge as the sample size increases; moreover, as shown in the previous section, the averaging approach yields larger improvements on the parameter estimates than on the AUC.

According to the estimates of the BN-EU model (under the averaging approach), uterine receptivity drops from 78% to 58% and eventually 26% for woman aged respectively {<34, 34-40, 40+}; moreover, embryo viability increases from 7% to 21% to 39% for embryos scored respectively as non-top, top and top-history. These findings show that embryo viability is generally a more critical factor than uterine receptivity, as it is generally accepted in the IVF literature; moreover they show a clear pattern of embryo viability as a function of the score, and of uterine receptivity as a function of the woman age.

As a last finding we present in Table 1 also the AUCs obtained by Bayesian network classifiers such as AODE, TAN and naive Bayes (NB); a

	BN-EU	AODE	TAN	NB
AUC ₀	74.1	74.8	73.0	75.2
AUC ₁	67.0	68.0	65.1	68.4
AUC ₂	83.4	81.6	79.6	80.1

Table 1: Comparison of different classifiers over the IIRM data set (average over 10 runs of 5-folds cross-validation).

presentation of these models can be found in (Webb et al., 2005). All these classifiers are induced on a data set which contains the same information provided to model BN-EU; however, the data set is complete as it does not model the EU assumption. In particular, it contains the following features: age of the woman; total number of transferred embryos; the number of non-top, top, and top-history embryos transferred. According to the terminology of (Roberts, 2007), such a data set is aggregated at the *recipient level*, rather than modeling the interaction between uterus and embryos. Despite such classifiers being learned on a complete data set, theirs AUCs are not significantly better than those of the BN-EU model. However, the BN-EU model is more interpretable and provides more insights to IVF experts than a traditional classifier. We thus share the viewpoint of (Roberts, 2007): “*Most clinical studies of embryo-level factors avoid the partial observability problem by either considering only patients with single embryo transfer, or by using a recipient-level aggregated measure. Modelling approaches which correctly incorporate the structure of the data are to be preferred both for the analysis of studies of embryo-level viability predictors, and in order to derive parameters which can be used in the utilization of such procedures.*”

5 Conclusions

We have proposed a novel BN model for modelling IVF transfer and a simple but effective averaging approach for improving both parameter estimates and predictive inferences. As a further step we plan to study, both from a medical and a statistical viewpoint, further covariates as parents of both U and the \mathcal{E} -nodes.

Acknowledgments

Research partially supported by CTI (Commission for Technology and Innovation) grant n. 9707.1 PFSL-LS, Swiss NSF grant no. 200020-132252 and the Hasler foundation grant n. 10030.

References

- D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.
- D.A. Morales, E. Bengoetxea, P. Larrañaga, M. García, Y. Franco, M. Fresnada, and M. Merino. 2008. Bayesian classification for the selection of in vitro human embryos using morphological and clinical data. *Computer Methods and Programs in Biomedicine*, 90(2):104–116.
- SA Roberts, WM Hirst, DR Brison, A. Vail, et al. 2010. Embryo and uterine influences on IVF outcomes: an analysis of a UK multi-centre cohort. *Human Reproduction*.
- Stephen A. Roberts. 2007. Models for assisted conception data with embryo-specific covariates. *Statistics in Medicine*, 26(1):156–170.
- RR Saith, A. Srinivasan, D. Michie, and IL Sargent. 1998. Relationships between the developmental potential of human in-vitro fertilization embryos and features describing the embryo, oocyte and follicle. *Human Reproduction Update*, 4(2):121–134.
- G. Santafé, J.A. Lozano, and P. Larrañaga. 2006. Bayesian model averaging of naive bayes for clustering. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(5):1149–1161.
- AL Speirs, A. Lopata, MJ Gronow, GN Kellow, and WI Johnston. 1983. Analysis of the benefits and risks of multiple embryo transfer. *Fertility and sterility*, 39(4):468.
- G.I. Webb, J.R. Boughton, and Z. Wang. 2005. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24.
- H. Zhou and C.R. Weinberg. 1998. Evaluating effects of exposures on embryo viability and uterine receptivity in vitro fertilization. *Statistics in medicine*, 17(14):1601–1612.

Decision analysis networks

Francisco Javier Díez

Dept. Artificial Intelligence. UNED. Madrid, Spain

Manuel Luque

Dept. Artificial Intelligence. UNED. Madrid, Spain

Caroline Leonore König

Dept. Artificial Intelligence. UNED. Madrid, Spain

Abstract

In this paper we introduce a new type of probabilistic graphical model, called decision analysis networks (DANs). Like influence diagrams (IDs), DANs are much more compact and easier to build than decision trees, and are able to represent conditional independencies. However, IDs are unable to represent many real-world problems, as they require a total ordering of the decisions and have difficulties to represent structural constraints, while DANs can easily represent those kinds of problems. We also show that DANs compare favorably with other formalisms proposed recently for modeling asymmetric decision problems.

1 Introduction

The two formalisms most widely used for the representation and analysis of decision problems are decision trees (DT) and influence diagrams (IDs) (Howard and Matheson, 1984). DTs have the advantage of almost absolute flexibility, but have three drawbacks: their size grows exponentially with the number of variables, they cannot represent conditional independencies, and they require in general a preprocessing of the probabilities (Howard and Matheson, 1984; Bielza et al., 2011). Even in cases with only a few chance variables, this preprocessing of probabilities is a difficult task. On the other hand, IDs have the advantages of being very compact, representing conditional independence, and using direct probabilities, and the drawback that they can only represent symmetric decision problems. However, most real-world problems are asymmetric: there is *structural asymmetry* when the value taken on by a variable restricts the domain of other variables, and there is *order asymmetry* when several ordering of the decisions are possible, for instance, when the decisions about

what tests to perform can be taken in any order (Jensen et al., 2006; Bielza et al., 2011).

Several formalisms have been proposed for representing and solving asymmetric decision problems—see Section 4. However, as far as we can tell, none of them has been used to build any real-world application, which may be a sign that none of them is completely satisfactory. For this reason, we present a new formalism, called *decision analysis networks* (DANs), which, in our opinion, can represent asymmetric problems more naturally.

The rest of the paper is structured as follows. First, we introduce the n -test problem, which will serve us to illustrate the properties of DANs and to compare different formalisms. Section 2 presents the definition of DANs, Section 3 explains how to convert any DAN into a DT, Section 4 compares DANs with other formalisms, Section 5 discusses some lines for future research, and Section 6 contains the conclusions.

Example 1. The n -test problem consists in deciding how to treat a patient that may suffer from a certain disease D . After an initial exam-

ination of the symptoms, the doctor may order one or several of n available tests, each one having a cost. It is assumed that each test can be performed only once and that the result of each test will be known immediately or, at least, before making a decision about other tests. In its simplest version, we assume that there is only one symptom, S , and all the variables are dichotomic, i.e., that the disease and the symptom are either present or absent, and the result of each test is either positive or negative.

An instance of this problem is the diagnosis of diabetes based on two tests: blood and urine, where BT (UT) represents the decision of whether performing the blood (urine) test and by B (U) represents the result of the test (Demirer and Shenoy, 2006).

2 Definition of a DAN

2.1 Graph and variables of a DAN

The set of variables of a DAN, \mathbf{V} , is formed by three disjoint subsets, \mathbf{C} (chance variables), \mathbf{D} (decisions), and \mathbf{U} (utilities): $\mathbf{V} = \mathbf{C} \cup \mathbf{D} \cup \mathbf{U}$. The first represent real-world properties that are not under the direct control of the decision maker, while the utilities represent his/her preferences. We denote the variables by capital letters and their values by lower-case letters. A bold upper-case letter denotes a set of variables and a bold lower-case letter denotes a configuration of them.

The graph of a DAN is an acyclic directed graph (ADG) such that each node represents a variable; for this reason we will speak of nodes and variables indifferently. We use the terms *link* and *arc* as synonyms. In this paper we will assume that utility nodes cannot have children.

The DAN in Figure 1 contains four chance variables, drawn with circles, three decisions, drawn with rectangles, and three utilities, drawn with diamonds.¹

It is possible to define a list of stages for a DAN, representing different phases of the decision process, such that each node belongs to

¹This DAN, encoded in the ProbModelXML format, can be found at www.cisiad.uned.es/ProbModelXML/examples and opened with OpenMarkov, an open-source tool available at www.openmarkov.org.

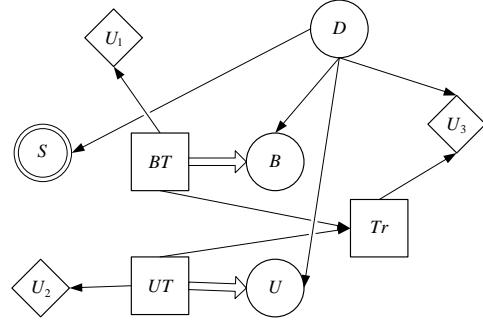


Figure 1: A DAN for the diabetes problem.

one stage. However, in this paper, due to space limits, we describe only one-stage DANs.

2.2 Restrictions

A *restriction* associated to a link $X \rightarrow Y$, where X and Y are chance variables or decisions, is a pair (x, y) , where x is a value of X and y is a value of Y . It means that variable Y cannot take the value y when X takes the value x . If X and Y are discrete, the restrictions associated with this link can be represented by a compatibility table with a column for each value of x and a row for each value of y ; the cell (x, y) contains a 1 when x and y are compatible and 0 when there is a restriction.² For example, Table 1 means that when the decision about blood test (BT) is not to perform it, the result (B) is neither positive nor negative.

BT		$+bt$	$-bt$
B	$+b$	1	0
	$\neg b$	1	0

Table 1: Compatibility table for the link $BT \rightarrow B$.

2.3 Potentials

A potential ψ is a function that maps each configuration \mathbf{x} of a set of variables \mathbf{X} onto a value

²In OpenMarkov, cells corresponding to compatible pairs are colored in green and those corresponding to restrictions are colored in red. Arcs with restrictions are marked with a short perpendicular double line. The compatibility table and the revelation conditions (see below) associated to a link can be accessed from its contextual menu.

of the set $\mathbb{R} \cup \{-\}$, i.e., $\psi(\mathbf{x})$ is either a real number or “ $-$ ”. Each chance variable Y whose parents in the graph are \mathbf{X} has an associated potential, denoted by $P(y|\mathbf{x})$, satisfying three conditions:

- $P(y|\mathbf{x}) \in [0, 1] \cup \{-\}$
- if there is a restriction (x_i, y) and $\mathbf{x}^{\downarrow X_i} = x_i$ (i.e., the i -th value of configuration \mathbf{x} is the same as x_i in the restriction), then $P(y|\mathbf{x}) = -$;
- for each configuration \mathbf{x} of \mathbf{X} , if some of the values of $P(y|\mathbf{x})$ are real-numbers, then their sum is 1.

Table 2 shows the potential for variable B , assuming that the sensitivity of this test for disease D is 0.92 and its specificity is 0.97.

BT	$+bt$		$\neg bt$	
D	$+d$	$\neg d$	$+d$	$\neg d$
B	$+b$	0.92	0.03	—
	$\neg b$	0.08	0.97	—

Table 2: Potential $P(b|bt, d)$, associated to node B .

2.4 Representing the flow of information

In DANs there are three ways to indicate when a variable becomes observed: always observed variables, revelation arcs, and stages.

2.4.1 Always observed variables

We can declare some chance variables as *always observed*, which means that we do not need to take any action to know their value. For instance, in Figure 1 variable S is put in a double circle, because we always know whether the patient has the symptom, without performing any test.

2.4.2 Revelation arcs

Given a link $D \rightarrow Y$, where D is a decision and Y a chance variable, we can declare that some values of D *reveal* the value of Y ; we then say that $D \rightarrow Y$ is a *revelation arc*. For instance, the revelation arc $BT \rightarrow B$ in

Figure 1, drawn as a double-line arrow, indicates that when we decide to perform the test ($BT = +bt$), its result becomes known for future decisions.

2.5 Summary: The meaning of links

In DANs a link $X \rightarrow Y$ may have five meanings.

1. *Causal influence*: when Y is a chance node.
2. *Functional dependence*: when Y is a utility node.
3. *Temporal order*: when both X and Y are decisions.
4. *Revelation*: when X is a decision and Y is a chance node.
5. *Restriction*: when X and Y are chance or decision nodes.

The first three meanings are the same as in IDs. Revelation arcs in DANs substitute information arcs in IDs.

3 Equivalent decision tree

In this section we explain how to convert a DAN into a DT, with the main purpose of providing a clear semantics for DANs. We denote each node in the tree by the variable that it represents, for instance X , and each branch departing from this node by its associated value, x .

The initial step is to pick up the always-observed variables, order them arbitrarily (the order will not affect the evaluation of the tree) and organize them into a tree: the first variable will be the root, there will be a branch for each value of this variable, each branch will contain a node for the second variable, and so on. In the diabetes problem, there is only one always-observed variable, S , which will be the root of the tree. It has two branches, $+s$ and $\neg s$, as shown in Figure 2.

Then we consider the order of the decisions. For each pair of decisions, if there is a directed path from D_1 to D_2 , then D_1 must be closer to the root in the tree. For example, the links $BT \rightarrow Tr$ and $UT \rightarrow Tr$ mean that the decisions BT and UT must be made before Tr , but the DAN does not specify which of BT and UT must be made first. Therefore, the decision maker faces one additional decision, that

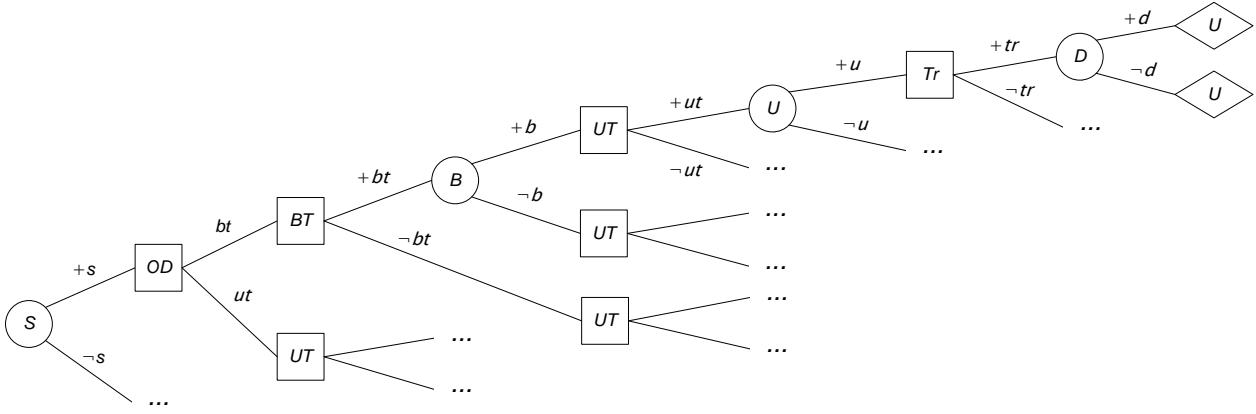


Figure 2: A decision tree for the DAN in Figure 1.

we denote by *OD* (order of decisions). For each branch of S , we add a decision node *OD* with two branches, labelled *bt* (which means that decision *BT* is made first) and *ut*. In the *bt* branch, we place a decision node *BT*, and in the *ut* branch a node *UT*.

If the value d of decision D reveals the value of some variables, these must appear immediately in the branch d . For example, given that $+bt$ reveals the value of B , we place in this branch a chance node B . For each branch of this node there are two remaining decisions, *UT* and *Tr*. As *UT* must be made first, we place a decision node *UT* in each branch. In the branch $-bt$ no variable is revealed.

Each of the three *UT* nodes has two branches, $+ut$ and $-ut$, which are expanded as in the case of *BT*. Next we place the last decision, *Tr*, which does not reveal the value of any variable, and finally we place in the tree the unobserved variables. In the upper scenario, the only unobserved variable is D . In the scenarios containing the $-bt$ branch shown in Figure 2, we should add also the variable B , but because of the restrictions in Table 1, no value of this variable is possible, which means that the variable B does not make sense in these scenarios, and therefore it must not appear in this part of the tree.

Finally, we place at each branch a leaf node, which represents a utility.

Each path from the root node to a leaf node defines a *scenario*. The probability of

each scenario is the product of the conditional probabilities involved in it, and its utility is the sum of the utility functions. For example, the probability of the upper scenario $(+s, bt, +bt, +b, +ut, +u, +tr, +d)$ is $P(+s|+d) \cdot P(+b|+d, +bt) \cdot P(+u|+d, +ut) \cdot P(+d)$ and its utility is $u_1(+bt) + u_2(+ut) + u_3(+tr, +d)$. Using the probabilities of the scenarios, it is possible to compute the conditional probability for each branch of the tree given the values of the variables at the left of that branch by normalizing them. This concludes the construction of the DT.

4 Comparison of different formalisms

In this section we examine seven formalisms for decision analysis. First we compare DANs with IDs, which are the standard probabilistic graphical model for decision analysis, and then (in Sec. 4.2) we analyze six formalisms for asymmetric decision problems.

4.1 DANs vs. IDs

There are many problems for which the DAN representation is much more simple than its ID counterpart. Let us consider, for instance, the one-test problem, in which the ID needs to introduce a dummy value *no-result* for the variable that represents the result of the test, which complicates the CPT for this variable, which will have 3 rows instead of 2. In the n -test prob-

lem, we need a decision node, T_1 , representing which test is performed in the first place, if any; therefore the domain of T_1 has $n + 1$ values. Then we need a chance variable, R_1 , for the result of the first test. Its probability table must contain two columns, $+d$ and $\neg d$, for each value of T_1 , which makes a total of $2(n + 1)$ columns and 3 rows. We need another decision node, T_2 , indicating which test is made in the second place. If we wish to indicate that the test performed first cannot be repeated, and that if the first decision is not to perform any test then the second one must also be not to test, we find a serious difficulty, because standard IDs do not admit restrictions. We might assign n values to T_2 , representing the $n - 1$ remaining test plus the possibility of no test, but the meaning of each value of T_2 would depend on the test performed in the first place. Therefore, the CPT for the result of the second test, R_2 , is difficult to build, and its numerical values are the same as some of those in the table for R_1 , which complicates the maintainability of the model and makes sensitivity analysis impossible, unless we introduce symbolic names for the parameters. In summary, the representation of the n -test problem with IDs is difficult for $n = 2$ and virtually unfeasible for $n > 2$.

This difficulty is due to one of the basic properties of IDs: they can only represent symmetric problems. In a few cases, asymmetric problems can be made symmetric by adding dummy states, such as *no-result*, but IDs are unsuitable for many asymmetric problems, such as the n -test problem, that can be easily represented with DANs.

In general, for each ID there is an equivalent one-stage DAN; this is the case for all the IDs we have found in the literature and in our practice. In the extremely rare cases in which there is no one-stage DAN for a certain ID, it is always possible to build an equivalent multi-stage DAN.

4.2 Comparison with other formalisms for asymmetric decision problems

In this section we compare briefly six formalisms for representing asymmetric decision

problems: influence diagrams with constraints (IDCs) (Smith et al., 1993), asymmetric influence diagrams (AIDs) (Nielsen and Jensen, 2000), sequential valuation networks (SVNs) (Shenoy, 2000; Demirer and Shenoy, 2006), unconstrained influence diagrams (UIDs) (Jensen and Vomlelová, 2002; Ahlmann-Olsen et al., 2009), sequential influence diagrams (SIDs) (Jensen et al., 2006), and decision analysis networks (DANs)—see also the comparison in (Bielza and Shenoy, 1999). Table 3 summarizes the main features of each method.

The first column indicates whether the method requires a total ordering of the decisions. The methods that impose this condition, such as IDCs, AIDs and SVNs, have to include decision nodes with as many states as the total number of decisions, as we explained in Section 4.1 when discussing why IDs cannot represent the n -test problem, and for the same reason, these formalisms are unsuitable for decision problems involving order asymmetries.

The second column means that IDs, IDCs, and UIDs need to add dummy states, such as *no-result*, to reflect the fact that the test result is not available when the test is not performed. This complicates the edition of the probability tables and the policies obtained.

The third column indicates which methods need to add labels to some arcs. In AIDs, the labels indicate which variable is revealed depending on which test has been performed. The problem is that a decision node representing the n tests will have n outgoing arcs, which complicates the representation when n is large. In the case of SVNs and SIDs, the labels indicate the sequence of decisions and observations, and again this makes the graph difficult to read for the n -test problem and for many others. In the case of a revelation arc $X \rightarrow Y$ in a DAN, the indication of which values of X reveal the value of Y can be seen as a label for that arc; however, this label only depends on X , not on any sequence of previous decisions and observations, and therefore the label is always very simple, independently of the complexity of the decision problem. In contrast, UIDs do not use labels because they assume that decisions are

	total order	dummy states	labeled links	inform. arcs
IDs (Howard and Matheson, 1984)	yes	yes	no	yes
IDCs (Smith et al., 1993)	yes	yes	no	yes
AIDs (Nielsen and Jensen, 2000)	yes	no	yes	yes
UIDs (Jensen and Vomlelová, 2002)	no	yes	no	no
SIDs (Jensen et al., 2006)	no	no	yes	yes
SVNs (Shenoy, 2000)	yes	no	yes	yes
DANs	no	no	no	no

Table 3: Main features of several methods for representing decision problems.

binary: Y is revealed if and only if X takes the value *true*. This is a limitation for non-binary decisions. For example, an echocardiogram can be transtoracic or transesophageal (decision X); the sensitivity and specificity of a finding, such as the calcification of a valve, depend significantly on the type of echocardiogram, but this difference cannot be modelled if the decision is modelled as a binary variable: do test / do not test.

Finally, the forth column shows which methods need information arcs, which are all except UIDs and DANs. Using these arcs forces the expert to indicate whether the variable is known or not to when making a decision, and complicates the graph when there are many always-observed variables and also when there are many sequences of decisions that can make a variable observed. In the n -test problem, we would need an arc from each result-of-test variable to each test decision. König (2012) has built an IDC, an AID, a UID, an SVN, and an SID for the diabetes problem. Seeing how cumbersome those diagrams are, even for a two-test problem, it is easy to understand why none of these methods is appropriate for the n -test problem.

In contrast, the graph of the UID for the diabetes problem is virtually the same as that of the DAN (see Figure 1), because none of these models require information arcs. This is a crucial advantage in many cases, such as the n -test problem, but turns out to be a problem in other situations, because both UIDs and one-stage DANs assume that the value of a variable is known immediately after making the decision

that reveals it. Multi-stage DANs overcome this difficulty, but UIDs do not.

In the same way, if any of two different decisions can reveal the value of a variable, this cannot be represented with an UID, either, because UIDs assume that a variable is known only when all its parent decisions have been made. (In contrast, in a DAN it suffices that one of the decisions is made.) This means that there are some problems that can be represented with IDs, but not with UIDs, which is a drawback of UIDs with respect to all the other formalisms listed in Table 3.

Finally, another drawback of UIDs, due to the lack of restrictions, is that they often require dummy states, which make the representation more obscure and less efficient.

A more detailed comparison of these formalisms for different decision problems can be found in (König, 2012).

5 Future work

In the paper that presented influence diagrams (Howard and Matheson, 1984) by the first time, the only evaluation method proposed was to build an equivalent decision tree. Later, several researchers developed more efficient methods, including algorithms for dealing with continuous variables. Similarly, in this paper we have introduced DANs and explained how to convert them into decision trees, which, given the power of today's computers, is sufficient for many real-world problems. In a future paper we will present more efficient algorithms. As an advance, we can say that symmetric DANs can be evaluated by variable elimination and arc

reversal, as in the case of IDs; structural asymmetries can be tackled by adding dummy nodes in the evaluation; order asymmetries can be solved by decomposing the problem into a set of completely ordered subproblems, as in (Demirer and Shenoy, 2006) or by building an S-DAG, as in (Ahlmann-Ohlsen et al., 2009; Luque et al., 2010). DANs containing numeric variables might be solved by adapting some of the algorithms for IDs (Cobb and Shenoy, 2008; Bielza et al., 2011).

Another pending task is to complete the implementation of DANs in *OpenMarkov*,³ an open-source tool for probabilistic graphical models. Currently this tool can be used to edit DANs, but it cannot yet evaluate them.

6 Conclusion

In this paper, we have proposed a new type of probabilistic graphical model, called *decision analysis networks* (DANs), and have explained how to convert each DAN into a decision tree. The main purpose of this conversion is to give a clear semantics to our model (in other models presented in the literature, the semantics is described only by an example). Any other algorithm should prove that it returns the same expected utility and optimal strategy as the evaluation of the tree.

A second reason for this conversion is that many decision analysts will understand DANs much better if they can examine the equivalent decision tree—in medicine, for example, the standard analysis tool are decision trees, while influence diagrams are almost unknown (Pauker and Wong, 2005), let alone the recent methods for asymmetric decision problems discussed in Section 4.2. In fact, the conversion of IDs into DTs was one of the explanation facilities proposed in (Lacave et al., 2007).

As explained in Section 4.1, for every influence diagram there is an equivalent DAN, but the converse is not true: there are many cases that are difficult if not impossible to represent with IDs, but can be modelled easily with DANs, such as the n -tests problem. Even

for problems that can be modelled with IDs, there is usually a DAN being more simple and more intuitive than the ID, because it does not need dummy states; an example is the one-test problem. The main reason why IDs have troubles to represent order asymmetry is the use of information arcs, as explained in that section, and therefore the other formalisms proposed recently for representing asymmetric decision problems face the same difficulty. The only exception are unconstrained influence diagrams (UIDs), which share with DANs the advantage of not using information arcs. However, unlike the rest of the models, UIDs have the drawback that for some IDs there is no equivalent UID. Another difference between UIDs and DANs is that the former do not use restrictions. It would be possible to extend UIDs with restrictions, stages, and labels for indicating which values of the decision reveal the value of a chance variable, but there would still be a subtle difference in the semantics of both types of models: in a DAN, every chance variable is revealed as soon as we decide to observe it, while in UIDs its value is not known until we make all the decisions about the actions that may reveal it.

Another difference between DANs and the rest of the models for asymmetric problems is that these have been illustrated with toy problems, and apparently they have not been used to build any real-world application. In contrast, we arrived at DANs when trying to solve an instance of the n -test problem: the mediastinal staging of non-small cell lung cancer. There are several tests available (CT-scan, PET, TBNA, EBUS, EUS, mediastinoscopy, etc.) and the experts do not agree on the optimal sequence of them. Clearly, similar problems are typical in other domains, such as troubleshooting different types of devices.

Additionally, it seems that none of those models has been implemented, with the exception of a program for evaluating UIDs that is not available outside the research group that used it for some experiments (Ahlmann-Ohlsen et al., 2009). In contrast, it is possible to edit DANs in *OpenMarkov* and store them in the Prob-

³www.openmarkov.org.

ModelXML format,⁴ and in the near future this tool will have algorithms for evaluating them.

Given the advantages of DANs over decision trees and influence diagrams and the fact that we have found so far no problem based on the non-forgetting assumption—which is also implicit in decision trees—that cannot be represented with DANs, we conjecture that this formalism will play a significant role as a tool for decision analysis.

Acknowledgments

We thank the anonymous reviewers of the PGM-2012 workshop for many useful comments.

This work has been supported by grants TIN2006-11152 and TIN2009-09158 of the Spanish Ministry of Science and Technology.

References

- [Ahlmann-Ohlsen et al.2009] K. S. Ahlmann-Ohlsen, F. V. Jensen, T. D. Nielsen, O. Pedersen, and M. Vomlelová. 2009. A comparison of two approaches for solving unconstrained influence diagrams. *International Journal of Approximate Reasoning*, 50:153 – 173.
- [Bielza and Shenoy1999] C. Bielza and P. P. Shenoy. 1999. A comparison of graphical techniques for asymmetric decision problems. *Management Science*, 45:1552–1569.
- [Bielza et al.2011] C. Bielza, M. Gómez, and P. P. Shenoy. 2011. A review of representation issues and modelling challenges with influence diagrams. *Omega*, 39:227–241.
- [Cobb and Shenoy2008] B. R. Cobb and P. P. Shenoy. 2008. Decision making with hybrid and influence diagrams using mixtures of truncated exponentials. *European Journal of Operational Research*, 186:261–275.
- [Demirer and Shenoy2006] R. Demirer and P. P. Shenoy. 2006. Sequential valuation asymmetric decision problems. *European Journal of Operational Research*, 169:286–309.
- [Howard and Matheson1984] R. A. Howard and J. E. Matheson. 1984. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, pages 719–762. Strategic Decisions Group, Menlo Park, CA.
- [Jensen and Vomlelová2002] F. V. Jensen and M. Vomlelová. 2002. Unconstrained influence diagrams. In *Proceedings of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI'02)*, pages 234–241, San Francisco, CA. Morgan Kaufmann.
- [Jensen et al.2006] F. V. Jensen, T. D. Nielsen, and P. Shenoy. 2006. Sequential influence diagrams: A unified asymmetry framework. *The International Journal of Approximate Reasoning (IJAR)*, 42:101–118.
- [König2012] C. König. 2012. Representing decision problems with Decision Analysis Networks. Master’s thesis, Dept. Artificial Intelligence, UNED, Madrid, Spain.
- [Lacave et al.2007] C. Lacave, M. Luque, and F. J. Díez. 2007. Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics*, 37:952–965.
- [Luque et al.2010] M. Luque, T. D. Nielsen, and F. V. Jensen. 2010. An anytime algorithm for evaluating unconstrained influence diagrams. Technical Report CISIAD-10-05, Dept. Artificial Intelligence, UNED, Madrid, Spain.
- [Nielsen and Jensen2000] T. D. Nielsen and F. Jensen. 2000. Representing and solving asymmetric Bayesian decision problems. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 416–425, San Francisco, CA. Morgan Kaufmann.
- [Pauker and Wong2005] S. Pauker and J. Wong. 2005. The influence of influence diagrams in medicine. *Decision Analysis*, 2:238–244.
- [Shenoy2000] P. P. Shenoy. 2000. Valuation network representation and solution of asymmetric decision problems. *European Journal of Operational Research*, 121:579–608.
- [Smith et al.1993] J. E. Smith, S. Holtzman, and J. E. Matheson. 1993. Structuring conditional relationships in influence diagrams. *Operations Research*, 41:280–297.

⁴www.cisiad.uned.es/ProbModelXML.

Gibbs sampling for parsimonious Markov models with latent variables

Ralf Eggeling¹, Pierre-Yves Bourguignon^{2,3}, André Gohr¹, Ivo Grosse¹

¹ Martin Luther University Halle-Wittenberg, Germany

² Max-Planck-Institute for Mathematics in the Sciences, Germany

³ Laboratoire de Physique Statistique, UMR 8550 CNRS/ENS/Univ. Paris VI and VII, France

Abstract

Parsimonious Markov models have been recently developed as a generalization of variable order Markov models. Many practical applications involve a setting with latent variables, with a common example being mixture models. Here, we propose a Bayesian model averaging approach for learning mixtures of parsimonious Markov models that is based on Gibbs sampling. The challenging problem is sampling one out of a large number of model structures. We solve it by an efficient dynamic programming algorithm. We apply the resulting Gibbs sampling algorithm to splice site classification, an important problem from computational biology, and find the Bayesian approach to be superior to the non-Bayesian classification.

1 Introduction

Assigning data points to classes based on their similarity to labeled training data is a pervasive task in almost all fields of science. One generally proceeds by assigning a probability to any observation X under the hypothesis that it belongs to some class k . It is customary to use the predictive probabilities $P(X|Y_k)$, which take different forms in the non-Bayesian and Bayesian settings, respectively. Both approaches assume X and training data Y_k to be generated from the same parametric statistical model \mathcal{M} . The non-Bayesian approach maps Y_k onto a unique parameter value by means of an estimator (ML, MAP, MP), and recycles this value for computing the predictive distribution

$$P_1(X|Y_k) = P(X|\hat{\theta}_{\mathcal{M}}(Y_k)), \quad (1)$$

whereas the Bayesian classification relies on the predictive distribution P_2 defined by the following integral over the parameter space:

$$P_2(X|Y_k) = \int P(X|\theta_{\mathcal{M}})P(\theta_{\mathcal{M}}|Y_k)d\theta_{\mathcal{M}}. \quad (2)$$

Here, the term Bayesian refers to the operation of averaging contributions from the whole

model, as opposed to the estimation of a single distribution that is further used for prediction purposes. The probability assignments P_1 and P_2 are not completely unrelated: With $\hat{\theta}_{\mathcal{M}}(Y_k)$ being the mode of one term in the integrand in Equation 2, P_1 can be understood as an approximation of P_2 , where only the parameter value contributing most is taken in consideration. P_2 , on the other hand, aggregates the contributions of all parameter values according to the support they provide to the training data Y_k .

In many applications, there is also uncertainty about the model \mathcal{M} . Viewing \mathcal{M} as a discrete-valued component of the parameter space, P_1 and P_2 are paralleled by P_3 and P_4 , respectively, in this setting:

$$P_3(X|Y_k) = P(X|\hat{\theta}_{\hat{\mathcal{M}}(Y_k)}(Y_k)), \quad (3)$$

which includes a model selection step for computing $\hat{\mathcal{M}}(Y_k)$, and

$$\begin{aligned} P_4(X|Y_k) &= \sum_{\mathcal{M}} P(\mathcal{M}) \\ &\cdot \int P(X|\theta_{\mathcal{M}})P(\theta_{\mathcal{M}}|Y_k)d\theta_{\mathcal{M}}, \end{aligned} \quad (4)$$

for the Bayesian classification, which yields a Bayesian model averaging task.

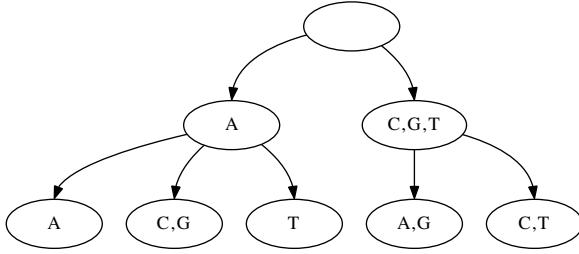


Figure 1: **Example PCT** of depth 2 over DNA alphabet. It encodes the partitioning of all 16 possible sequences into the subsets $\{AA\}$, $\{CA, GA\}$, $\{TA\}$, $\{AC, AG, AT, GC, GG, GT\}$, and $\{CC, CG, CT, TC, TG, TT\}$.

In both cases, the choice of an appropriate set of candidate models is crucial. Here, we focus on parsimonious Markov models, which have been introduced by Bourguignon (2008) as an extension of variable order Markov models (Rissanen, 1983; Bühlmann and Wyner, 1999). They use parsimonious context trees (PCTs), which differ from traditional context trees in two aspects: (i) a PCT is a balanced tree, i.e. each leaf has the same depth, and (ii) each child node represents an arbitrary subset of the alphabet \mathcal{A} , with the additional constraint that sibling nodes form together a partition of \mathcal{A} . An example PCT, which forms a partition of strings that cannot be represented by a traditional context tree, is shown in Figure 1.

Many practical applications involve a setting with latent variables or incomplete training data, for example Hidden Markov Models (Rabiner, 1989) or mixture of trees (Meila and Jordan, 2000). In such cases, classification is based on the incomplete versions of P_1 to P_4 , i.e. averages thereof against the latent variables. The associated enumeration of all realizations of the latent variables cannot, in general, be performed in an exact yet computationally inexpensive way. As an alternative, the EM algorithm (Dempster et al., 1977) is often used to derive an approximation of $\hat{\theta}_{\mathcal{M}}$. In the Bayesian setting, another option is offered by algorithms that generate samples drawn (at least approximately) from the posterior distribution of the parameter, $P(\theta_{\mathcal{M}}|Y_k)$. Predictive probabilities

such as P_2 and P_4 can then be derived by substituting a discrete approximation of the integral using the generated parameter values.

One of the simplest models that uses latent variables is a C -component mixture model. A recent algorithm for learning mixtures of inhomogeneous parsimonious Markov models is based on the MAP principle and thus uses a modified EM algorithm (Gohr et al., 2012). Here, we derive a Gibbs sampler for sampling from the posterior distribution of mixtures of inhomogeneous parsimonious Markov models. We study the convergence behavior of the algorithm and evaluate its classification performance compared to the corresponding EM algorithm.

2 Model and prior

The data are symbolic sequences of fixed length L over the alphabet \mathcal{A} . We denote a single symbol by X , a sequence of length L by $\vec{X} = (X_1, \dots, X_L)$ and a data set of N sequences by $\mathbf{X} = (\vec{X}_1, \dots, \vec{X}_N)$.

2.1 Parsimonious Markov model

Here, we focus on inhomogeneous parsimonious Markov models (parsMMs) of order D , which use potentially different *parsimonious context trees* (PCTs) for each position. A PCT is a rooted, balanced tree, which we subsequently denote by τ . $\mathcal{T}_D^{\mathcal{A}}$ is the set of all possible trees for a given alphabet \mathcal{A} and depth D . Each node of a PCT is labeled with a non-empty subset of \mathcal{A} , except for the root, which is labeled by the empty subset. The set of labels of all children of an arbitrary inner node forms a partition of \mathcal{A} . Starting from a leaf, building the cartesian product of all subsets found on the path to the root defines a set of sequences, which is the *context* encoded by that leaf.

The example PCT in Figure 1 encodes the contexts $\{A\} \times \{A\}$, $\{C, G\} \times \{A\}$, $\{T\} \times \{A\}$, $\{A, G\} \times \{C, G, T\}$, and $\{C, T\} \times \{C, G, T\}$.

A parsimonious Markov model of order D for a sequence \vec{X} of length L involves exactly L PCTs, denoted by $\vec{\tau} = (\tau_1, \dots, \tau_L)$. For the ease of presentation, we exclude from the following discussion the first D PCTs, which have an increasing maximal depth of $0, \dots, D-1$. We

denote a single context as \mathbf{w} , and all contexts represented by a specific parsimonious context tree τ by \mathcal{C}_τ . For a given PCT τ , we denote the conditional probability of observing a symbol $a \in \mathcal{A}$, given that the concatenation of the preceding D symbols is in \mathbf{w} , as $\theta_{\mathbf{w}a}^\tau$. We denote all parameters of a single position in the parsMM by $\Theta = (\tau, (\vec{\theta}^{\tau'})_{\tau' \in \mathcal{T}_D^\mathcal{A}})$, using the product space formulation of Carlin and Chib (1995) in order to ensure a fixed dimensionality of the parameter space. We further combine the parameters of all positions in the parsMM by $\vec{\Theta} = (\Theta_1, \dots, \Theta_L)$. The likelihood function of a parsMM is given as

$$P(\mathbf{X}|\vec{\Theta}) = \prod_{\ell=1}^L \prod_{\mathbf{w} \in \mathcal{C}_{\tau_\ell}} \prod_{a \in \mathcal{A}} (\theta_{\ell \mathbf{w} a}^{\tau_\ell})^{N_{\ell \mathbf{w} a}}, \quad (5)$$

where $N_{\ell \mathbf{w} a}$ is the number of occurrences of symbol a at position ℓ in all sequences in data set \mathbf{X} where the symbols from position $\ell - D$ to $\ell - 1$ are an element of \mathbf{w} . We define a prior for the parsimonious Markov model by

$$P(\vec{\Theta}) = P(\vec{\tau}) \prod_{\ell=1}^L \prod_{\tau' \in \mathcal{T}_{\ell,D}^\mathcal{A}} \prod_{\mathbf{w} \in \mathcal{C}_{\tau'}} P(\vec{\theta}_{\ell \mathbf{w}}^{\tau'}) \quad (6)$$

where $P(\vec{\theta}_{\ell \mathbf{w}}^{\tau'})$ is a Dirichlet distribution with hyperparameters $\vec{\alpha}_{\ell \mathbf{w}}^{\tau'}$. For the case studies in this work, we further restrict the parameter prior to a symmetric Dirichlet distribution. Following the equivalent sample size (ESS) concept (Heckerman et al., 1995), we obtain a natural computation of the pseudocounts from the ESS that is inspired by Bayesian networks, namely $\alpha_{\ell \mathbf{w} a}^{\tau'} = \frac{\text{ESS}[\mathbf{w}]}{|\mathcal{A}|^{D+1}}$. We specify the structure prior over all L PCTs in the model by

$$P(\vec{\tau}) \propto \prod_{\ell=1}^L \kappa^{|\mathcal{C}_{\tau_\ell}|}. \quad (7)$$

It depends on one hyperparameter $\kappa \in (0, \infty)$, which can be used to influence the number of leaves and thus the complexity of the model, interpolating between the two special cases: When $\kappa \rightarrow +\infty$, the maximal tree, which represents a full order Markov model, receives a prior

probability of one. Conversely, when if $\kappa \rightarrow 0$, only the minimal tree, which represents an independence model, receives prior support.

2.2 Mixture model

We consider a C -component mixture model as a typical instance of a model with latent variables. The assignment of each of the N sequences to one of the C components is specified by the latent vector $\vec{u} = (u_1, \dots, u_N) \in \{1, \dots, C\}^N$. Each component of the mixture model is an inhomogeneous parsimonious Markov model, so the complete set of parameters is denoted by $\Theta = (\vec{\pi}, \vec{\Theta}_1, \dots, \vec{\Theta}_C)$, where $\vec{\pi} = (\pi_1, \dots, \pi_C)$, and π_c contains the probability of the c -th mixture component. The likelihood of the mixture model is thus given as

$$P(\mathbf{X}, \vec{u}|\Theta) = P(\mathbf{X}|\vec{u}, \Theta)P(\vec{u}|\Theta), \quad (8)$$

where

$$P(\vec{u}|\Theta) = \prod_{i=1}^N \pi_{u_i}, \quad (9)$$

and

$$P(\mathbf{X}|\vec{u}, \Theta) = \prod_{c=1}^C P(\mathbf{X}_{\{\vec{u}=c\}}|\vec{\Theta}_c) \quad (10)$$

with

$$\mathbf{X}_{\{\vec{u}=c\}} = (X_i)_{i|u_i=c} \quad (11)$$

denoting all sequences that are assigned to component c by the latent variables \vec{u} . The prior of the full mixture model factorizes as

$$P(\Theta) = \prod_{c=1}^C P(\vec{\Theta}_c). \quad (12)$$

For the sake of simplicity, we set $\vec{\pi}$ in this work externally to a uniform distribution.

3 Gibbs sampling

We now shall generate a sample from the distribution $P(\Theta, \vec{u}|\mathbf{X})$, from which $P(\Theta|\mathbf{X})$ for use in Equation 4 can be further derived by marginalization.

Sampling from $P(\Theta, \vec{u}|\mathbf{X})$ directly is intractable, but approximate sampling techniques

$$\forall_{c=1}^C \forall_{\ell=1}^L : \text{sample } \tau_{cl}^{(t)} \text{ from } P(\tau_{cl} | \vec{u}^{(t-1)}, \mathbf{X}) \quad (13)$$

$$\forall_{i=1}^N \forall_{\ell=1}^L \forall_{w \in \mathcal{C}_{\tau_{cl}}} : \text{sample } \theta_{clw}^{\tau_{cl}(t)} \text{ from } P(\theta_{clw}^{\tau_{cl}} | \tau_{cl}^{(t)}, \vec{u}^{(t-1)}, \mathbf{X}) \quad (14)$$

$$\forall_{i=1}^N : \text{sample } u_i^{(t)} \text{ from } P(u_i | \Theta^{(t)}, \vec{X}_i) \quad (15)$$

Figure 2: **Sampling steps** for the t -th iteration of the Gibbs sampler.

are available. We focus here on Gibbs sampling (Geman and Geman, 1984; Casella and George, 1992), where each parameter is iteratively updated using its conditional distribution given the current value of the other parameters. Samples form a realization of a Markov chain, whose stationary distribution is the posterior.

We sample in the t -th iteration $\Theta^{(t)}$ from $P(\Theta | \vec{u}^{(t-1)}, \mathbf{X})$ and $\vec{u}^{(t)}$ from $P(\vec{u} | \Theta^{(t)}, \mathbf{X})$. The conditional probability of the parameters given the latent variables decomposes to

$$P(\Theta | \vec{u}^{(t-1)}, \mathbf{X}) = \prod_{c=1}^C \prod_{\ell=1}^L P(\tau_{cl}, \vec{\theta}_{cl} | \vec{u}^{(t-1)}, \mathbf{X}). \quad (16)$$

For each component c and each position ℓ (and thus omitting here both indices for the sake of convenience), we use the idea of variable grouping (Liu, 2001) and sample $(\tau, \vec{\theta}^\tau)^{(t)}$ jointly from $P(\tau, \vec{\theta}^\tau | \vec{u}^{(t-1)}, \mathbf{X})$ instead of sampling from each single conditional distribution separately. This is achieved in a hierarchical manner, by decomposing the joint conditional probability distribution into $P(\tau | \vec{u}^{(t-1)}, \mathbf{X})P(\vec{\theta}^\tau | \tau, \vec{u}^{(t-1)}, \mathbf{X})$. We first sample $\tau^{(t)}$ w.r.t. its conditional distribution given $(\vec{u}^{(t-1)}, \mathbf{X})$, and then $\vec{\theta}^\tau^{(t)}$ from its full conditional distribution, $P(\vec{\theta}^\tau | \tau^{(t)}, \vec{u}^{(t-1)}, \mathbf{X})$, under which the components of $\vec{\theta}^\tau$ are independent. Similarly, the components of \vec{u} are conditionally independent given the other parameters and the data.

As a result, we obtain the sampling scheme that is shown in Figure 2. The remaining task is to efficiently sample from the specified conditional distributions, which we discuss in the following sections. Whereas sampling of latent variables and probability parameters are com-

paratively simple, the particular challenge lies in the sampling of the PCTs.

3.1 Structure sampling

In this section, we focus on the sampling of a PCT structure in order to perform the sampling step 13. The probability of a particular tree structure given data (step 13) is

$$P(\tau_{cl} | \vec{u}, \mathbf{X}) \propto \prod_{w \in \mathcal{C}_{\tau_{cl}}} \kappa \frac{\mathcal{B}(\vec{N}_{clw} + \vec{\alpha}_{clw})}{\mathcal{B}(\vec{\alpha}_{clw})}, \quad (17)$$

where \mathcal{B} denotes the multinomial beta-function. Hence, the probability decomposes into a product of scores for each context, i.e. leaf scores for each leaf in the PCT. Each leaf score is itself a marginal likelihood for the particular context multiplied with the structure prior hyperparameter κ . While the probability for a given tree can be computed easily, the challenge lies in sampling one out of a super-exponential number (with respect to model order and alphabet size) of possible PCTs, without computing the probability for every single tree explicitly.

To this extent, we propose a dynamic programming algorithm, which is inspired by that of Bourguignon (2008) and Wolf and Willems (1994) for finding the PCT or CT that maximizes a given score.

The algorithm runs on a specific data structure, the extended tree, of which one subtree is shown in Figure 3. In contrast to a PCT, the children of a node of an extended tree do not form a partition of \mathcal{A} , but rather encompass all elements of $\mathcal{P}(\mathcal{A}) \setminus \{\emptyset\}$.

We construct top-down an extended tree of depth D , and then sample in a bottom-up traversal a regular PCT by taking – for each node n – one out of two possible actions:

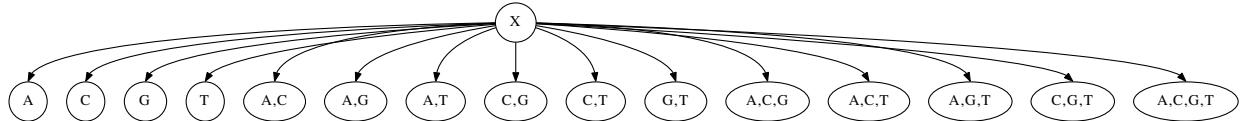


Figure 3: **Extended tree.** The picture depicts the structure of an arbitrary inner node (labeled with X) and its children in the extended tree over a four letter alphabet. The labels of all children of a node form the power set of \mathcal{A} , with the exception of the empty set.

(i) If n is a leaf (representing context \mathbf{w}), we compute the score $\kappa \frac{\mathcal{B}(\vec{N}_{cl\mathbf{w}} + \vec{\alpha}_{cl\mathbf{w}})}{\mathcal{B}(\vec{\alpha}_{cl\mathbf{w}})}$, and assign it to n .

(ii) If n is an inner node, we first compute the probability of each *valid choice* of children of n , where a valid choice is a set of children, whose labels form a partition of \mathcal{A} and the score of a valid choice is simply the product of the scores of the children contained in it. Next, one valid choice is sampled according to the computed probability distribution. The probability of this sampled set of children becomes the score of n . The remaining children of n , which do not belong to the sampled set, and all subtrees below are discarded. Hence n becomes the root of a subtree that satisfies the characteristics of a PCT and has a score assigned to it.

We obtain a complete PCT, once we have sampled a valid choice of children of the root of the extended tree.

The algorithm samples correctly from the posterior distribution for the following reasons: First, when sampling the children of a particular node, the scores of all potential children are already available. In step (ii), we can safely assume that for each inner node n , each child (Figure 3) is either a leaf, in which case we have obtained its score by step (i), or the root of a subtree that already satisfies the characteristics of a PCT and has a score assigned to it. Second, the subtree rooted at an arbitrary node n and subtrees rooted at the siblings of n are conditionally independent given the labels on the path from n to the global root of the PCT. This information is available at any time due to the top-down construction of the extended tree.

The time complexity of the algorithm is given by the number of valid choices multiplied by

the number of inner nodes in the extended tree, more precisely $\mathcal{O}\left(B_{|\mathcal{A}|}(2^{|\mathcal{A}|}-1)^{D-1}\right)$, where $B_{|\mathcal{A}|}$ is the Bell number (Rota, 1964).

3.2 Parameters

The probability of the conditional probability parameters of a given context of a given PCT structure (step 14) is a Dirichlet distribution with the hyperparameters being a sum of counts and pseudocounts, that is,

$$P(\theta_{cl\mathbf{w}}^{\tau_{cl}} | \tau_{cl}, \vec{u}, \mathbf{X}) = \text{Dir}(\theta_{cl\mathbf{w}}^{\tau_{cl}} | \vec{N}_{cl\mathbf{w}} + \vec{\alpha}_{cl\mathbf{w}}). \quad (18)$$

3.3 Latent variables

The conditional probability distribution of the latent variables (step 15) is merely a fraction of likelihood values:

$$P(u_i | \Theta, \vec{X}_i) = \frac{\pi_{u_i} P(\vec{X}_i | \vec{\Theta}_{u_i})}{\sum_{c=1}^C \pi_c P(\vec{X}_i | \vec{\Theta}_c)} \quad (19)$$

It is noteworthy that given the parameters and the data, the components of \vec{u} are mutually independent. In addition, the conditional distribution of each component u_i depends on the data only through the sequence X_i .

4 Case studies

In order to evaluate the method on real world data, we apply it to the problem of splice site classification, which is an important task in computational biology. Splice sites are short DNA sequences that contain a GT dinucleotide. However, not all sequences containing a GT dinucleotide are functional splice sites. The statistical problem is to distinguish functional splice sites from non splice sites by modeling the variable nucleotides left and right of the GT.

Table 1: **Data sets** used for the classification experiment, consisting of experimentally verified positive and negative data.

	training	test
splice donor sites	pos_train	pos_test
non splice sites	neg_train	neg_test

We use the splice donor site data sets of Yeo and Burge (2004), which consist of experimentally verified splice donor sites and non splice sites (Table 1). The conserved GT dinucleotide has been removed already, the remaining sequences of length of $L = 7$ over $\mathcal{A} \in \{A, C, G, T\}$ are diverse, which makes the classification problem challenging (Yeo and Burge, 2004). In the following, we use a mixture of $C = 2$ parsMMs of depth $D = 2$ with an ESS = 16 for each component for learning from the functional splice sites.

4.1 Convergence

Gibbs sampling, like any MCMC method, samples from the target distribution only asymptotically. It is therefore important to study the convergence rate of the algorithm in order to control the quality of the approximation brought by the sample. In practice, a number of first samples is generated but discarded, thereby skipping the burn-in phase of the sampler. We investigate the convergence for a data set of size $N = 500$, since we use data sets of the same size for a classification study in section 4.2. After initializing each latent variable by sampling from the uniform distribution (0.5, 0.5), we perform 10^4 iterations of the Gibbs sampler. In each iteration step, we store the sampled value of each of the 500 latent variables. In absence of a simple notion of correlation among PCTs, we focus here on the number of leaves of the PCTs. The tree of the first position of both components can be neglected, since it always consists of one leaf. So we store only the number of leaves of each PCT at position 2-7 in both components, yielding 12 additional variables per iteration, which makes 512 variables in total. Since the space of actively used proba-

bility parameters changes from iteration step to iteration step, we do not measure their convergence behavior.

Next, we compute the autocorrelation function for each of the 512 variables with lags of 1 to 500 from the 10^4 iterations. We repeat the process 10^2 times with different initializations and average the autocorrelation coefficients for each variable. We project the results to six curves in the following way. First, we investigate the latent variables and the PCT complexity separately. Second, we plot for each lag the (i) maximum, (ii) mean, and (iii) median of the absolute autocorrelation, which is shown in Figure 4. In both cases, we observe an approximately exponential decay of the autocorrelation up to a lag of approximately 150. For larger lags, it differs only slightly, even though there is a small increase between lag 250 and 350. We finally conclude that 200 iteration steps is the minimal length of the burn-in phase, since all samples before that are still correlated to the random initialization. For the following classification studies, we use a burn-in phase of 1000 iteration steps.

4.2 Classification

After having evaluated convergence behavior of the Gibbs sampling algorithm, we study how well it performs in practice compared to an EM algorithm when facing the task of classification. We intent to classify splice donor sites (positive) against non splice sites (negatives). To this extend, we use an independence model for the negative class. In the following study we use all four data sets that are depicted in Table 1. All data sets except **pos_train** are kept fixed during the entire experiment. **pos_train**, which we utilize to train the mixture model, is a random sample of 500 sequences from the original training data set of Yeo and Burge (2004).

Using these four data sets, we train a classifier by (a) training an independence model on **neg_train** and (b) training a two-component mixture of parsimonious Markov models on **pos_train** by (i) Gibbs sampling and (ii) the EM algorithm. In case of (i), we obtain a series of 10^3 parameter sets that are samples from the posterior distribution after the burn-in phase,

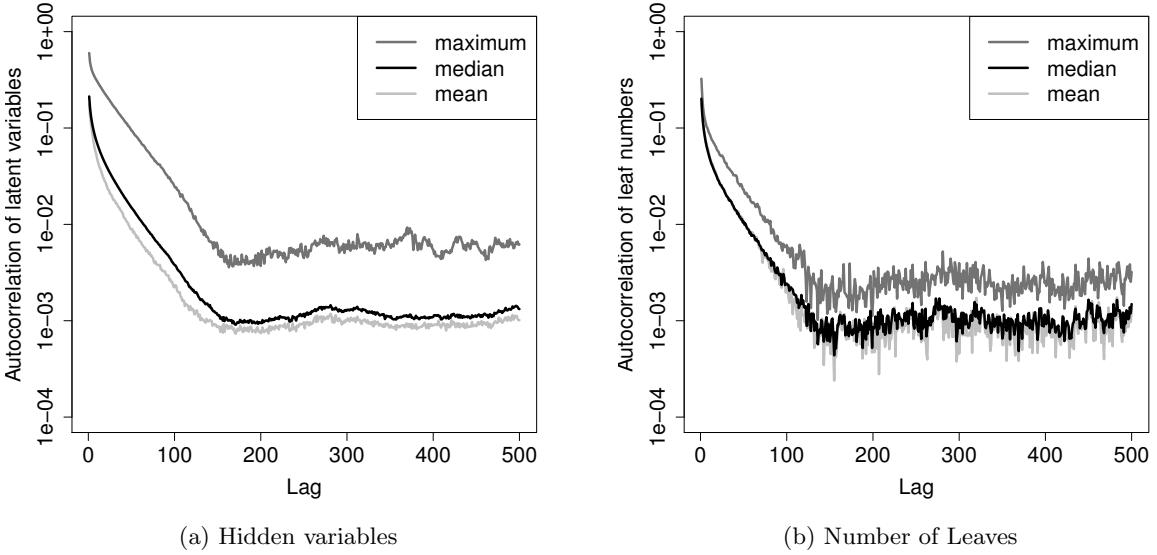


Figure 4: **Autocorrelation coefficients** of latent variables and number of leaves of the parsimonious context trees in logarithmic scale. Figure 4a depicts for each lag the mean, median and maximum of the autocorrelation over the 500 latent variables. Figure 4a depicts the same statistics of the autocorrelation of the PCT leaf number over the 12 nontrivial PCTs in the mixture model. In all cases, the autocorrelation shows a nearly exponential decay with a decay constant of approximately 1.7 until it remains stable at low values at lag 150-200.

where in only every hundredth iteration a parameter set is stored. We start the Gibbs sampler ten times and use all the 10^4 parameter sets for a classification according to Equation 4. In case of (ii), we run the EM algorithm of Gohr et al. (2012) until the difference in the posterior is smaller than 10^{-6} . We repeat the procedure ten times with different initializations, and use the parameter set from the run that yields the highest posterior in order to classify according to Equation 3. In both cases, we classify all sequences in `pos_test` and `neg_test` and compute the area under the ROC curve (AUC). We repeat the entire procedure with 20 different samples of size 500 from the training data set, and compute the mean AUC and its standard error.

The classification performance might be heavily influenced by the model complexity. So we repeat the study with different values of the structure prior constant κ . By varying κ from 10^{-50} to 10^{10} , we cover the whole scale of complexity that a parsMM(2) can represent, from

independence model, obtained when all PCTs contain only one leaf, to a second order inhomogeneous Markov model, obtained when all PCTs have the maximal number of leaves.

The results are shown in Figure 5. As we expected, the classification performance depends heavily on the complexity. However, for all levels of complexity, the Bayesian classification based on Gibbs sampling outperforms the non-Bayesian classification that uses the EM algorithm by a wide margin. Moreover, the standard errors are substantially smaller, indicating that the Bayesian classification also yields more stable results.

4.3 Conclusions

We derived a Gibbs sampling algorithm that samples PCT structures and corresponding probability parameters from the posterior distribution of parsimonious Markov models in a setting with latent variables. We studied the convergence of the algorithm and found 1000

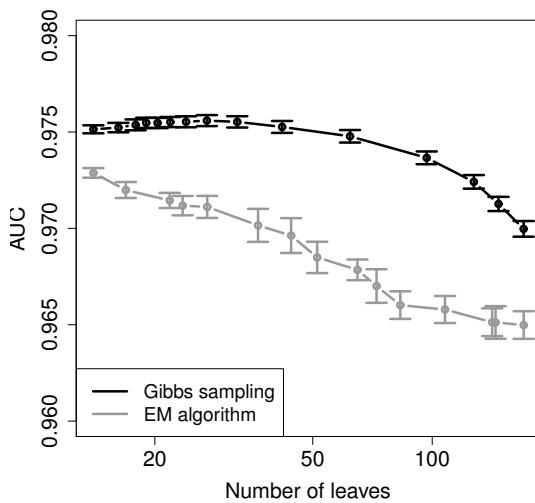


Figure 5: **Mean AUC** of a repeated holdout classification experiment for Gibbs sampler and EM algorithm for different model complexities. Each error bar shows the double standard error. Gibbs sampler outperforms EM algorithm for all possible model complexities.

sampling steps to be a safe estimation of the length of the burn-in phase. We applied the algorithm to splice site classification and observed the Gibbs sampler outperforming the EM algorithm for every model complexity. These results suggest that the Bayesian learning approach might also be useful when parsimonious Markov models are combined with other models that also involve latent variables.

Acknowledgments

This work was funded by *Reisestipendium des allg. Stiftungsfonds der MLU Halle-Wittenberg, MPG/CNRS SysBio research program*, and DFG (grant no. GR-3526_1-1). We thank Petri Myllymäki, Teemu Roos, and Antti Honkela for valuable discussions.

References

Pierre-Yves Bourguignon. 2008. *Parcimonie dans les modèles markoviens et applications à l'analyse des séquences biologiques*. Ph.D. thesis, Université Evry Val d'Essonne.

- P. Bühlmann and A.J. Wyner. 1999. Variable length Markov chains. *Annals of Statistics*, 27:480–513.
- B.P. Carlin and S. Chib. 1995. Bayesian Model Choice via Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(3):473–484.
- G. Casella and E.I. George. 1992. Explaining the Gibbs Sampler. *The American Statistician*, 46:167–174.
- A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.
- S. Geman and D. Geman. 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 6, pages 721–741.
- A. Gohr, S. Posch, and I. Grosse. 2012. Mixtures of Parsimonious Markov Models. Technical Report 2012/2, Institute of Computer Science, Martin Luther University Halle-Wittenberg, Germany.
- G. Heckerman, D. Geiger, and D. Chickering. 1995. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20:197–243.
- Jun S. Liu. 2001. *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics.
- M. Meila and M.I. Jordan. 2000. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48.
- L.R. Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–268.
- Jorma Rissanen. 1983. A universal data compression system. *IEEE Trans. Inform. Theory*, 29(5):656–664.
- G.C. Rota. 1964. The Number of Partitions of a Set. *Amer. Math. Monthly*, 71:498–504.
- P. Volf and F. Willem. 1994. Context maximizing: Finding MDL decision trees. In *15th Symp. Inform. Theory Benelux*, pages 192–200, May.
- G. Yeo and C.B. Burge. 2004. Maximum Entropy Modeling of Short Sequence Motifs with Applications to RNA Splicing Signals. *Journal of Computational Biology*, 11(2/3):377–394.

A framework for development, teaching and deployment of inference algorithms

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

We present **symfer**, a software framework for probabilistic inference algorithms. Each inference algorithm (like variable elimination, junction tree propagation, recursive conditioning) is represented as a symbolic manipulation of factor algebra expressions. In combination with the readability and terseness of Python code, this uniform representation makes the framework very suitable for teaching, as well as for explorative research: using the interactive Python interpreter, one can combine features of different algorithms and examine their effect. Numeric evaluation happens in a separate stage, implemented in Java/C for efficient execution and high portability. We exploit the latter feature in an application that performs inference on a smartphone.

1 Introduction

In popular software tools for inference in probabilistic graphical models such as Hugin, GeNIe/SMILE, SamIAM and the MATLAB Bayes Net Toolbox, an inference algorithm (such as variable elimination (Zhang and Poole, 1996), junction tree propagation (Shenoy and Shafer, 1988), Kalman filtering (Kalman, 1960)) is a mostly opaque process. The user supplies a model, a query and evidence and calls the algorithm; the algorithm runs and returns a probability distribution. Processes such as determining a graph triangulation, elimination order or junction tree often remain hidden, and options for influencing this process are limited.

Of course, this is exactly what many users expect from these tools; the decisions made in said processes form a layer of complexity from which they want to be shielded. On the other hand, *researchers* and *teachers* of probabilistic inference methods have contrary requirements: what they desire from software is easy inspection and full control of these processes.

The software framework **symfer** that we present in this article caters to these needs using a novel architecture: it splits up the inference process into a *symbolic* and a *numeric* stage. The symbolic stage contains high-level inference algorithms; it is

programmed in Python, which represents the algorithms in a uniform way which is readable (close to the pseudocode usually found in scientific literature) and easy to adapt. Moreover, the Python interpreter provides an interactive interface suited to explorative research similar to MATLAB.

The second stage performs the actual ‘number grinding’. It has very few dependencies on external libraries, which makes it easily portable to other platforms. We have exploited this ourselves by including it in a telemedicine application on an Android phone (van der Heijden et al., 2011), which performs inference in a medical probabilistic model. We expect that this easy portability will also interest other software developers.

2 Architecture

Figure 1 shows the basic architecture of our software. Almost everything of interest happens in the first stage. It consists of a Python library with an API for creating and manipulating *factor algebra* expressions (defined in section 3). These can be based on externally created models (currently we support only the Hugin format) or constructed from scratch in Python. The latter option might be of interest if the model has a repetitive structure, or needs to be generated from a higher level description.

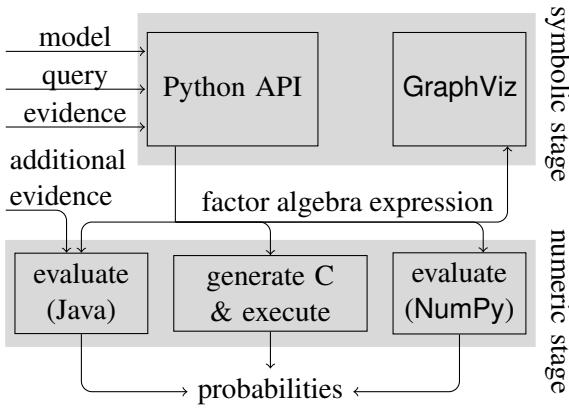


Figure 1: The two-stage architecture of `symfer`.

The purpose of this symbolic stage is to construct a factor algebra expression, which defines how to perform inference for a specific model, query and evidence combination. For this, the library includes several inference algorithms, of which the source code is meant for easy inspection and adaptation. The user can also use the API to write their own inference algorithms.

Subsequently, there are alternative ways to proceed. To obtain the resulting probabilities, the factor algebra expression (constructed in Python memory) can be evaluated either using Java, Python or C. For the first option, the expression is serialized to YAML (a human-readable textual format similar to JSON) and interpreted by a Java module, which evaluates it numerically. Optionally, the user can supply additional evidence $E=e$ to this module: an index operation $[E=e]$ is then inserted at the appropriate leaves, and the summation operation Σ_E is removed (we refer to section 3 for explanation of the operations). This option is practical for scenarios where the same query is repeated several times with different (maybe cumulative) evidence; in these cases, the first stage only needs to be applied once. The Java module is easily ported to other platforms; its only external dependency is the YAML parser.

Evaluation using C is different: dedicated C source code is generated for each operation in the factor algebra expression. This code can then be compiled and executed. The C code is faster, more predictable ('manual' garbage collection), and even

more portable than the Java evaluator, but introducing additional evidence is impossible. The third option, evaluating directly from Python (using array processing library NumPy), is easiest to use but not recommended for memory-intensive queries, as garbage collection seems to be inefficient.

To aid in teaching and exploration, the expression can also be exported to GraphViz (more specifically, dot) in order to show it as a tree (as in Figure 5).

3 Background: factor algebra

The symbolic expressions created using `symfer` consist of operations on *factors*. These operations are widely used in inference algorithms, e.g. see (Zhang and Poole, 1996) and (Koller and Friedman, 2009), but their symbolic use, although going back at least to (Shachter et al., 1990), is not widespread.

A factor is much like a mathematical function with multiple arguments, but refers to these arguments by name instead of by position, i.e. $f(X=x, Y=y)$ instead of $f(x, y)$. Thus, the order does not matter: $f(X=x, Y=y)$ is the same expression as $f(Y=y, X=x)$. In the scope of this article, a name like X refers to a probabilistic variable with a finite domain, written $\text{dom}(X)$. Formally, a factor is defined as a single-argument function on instantiations; an *instantiation* $\mathbf{v} = \{V_1=v_1, \dots, V_n=v_n\}$ is a function mapping each variable V_i to a value $v_i \in \text{dom}(V_i)$.

A factor defines a specific set of variables \mathbf{V} for which the instantiation must provide values; with a little abuse of terminology this set is called the factor's *domain* and written as $\text{dom}(f) = \mathbf{V}$. Values of a factor are written $f(\mathbf{v})$ or $f(V_1=v_1, \dots, V_n=v_n)$, where set braces are omitted for legibility, and are often probabilities. In `symfer`'s numeric stage, a factor is implemented as an *array* of all its function values, with size $\prod_{V_i \in \text{dom}(f)} |\text{dom}(V_i)|$.

Factor algebra provides the tools for manipulating factors:

- *Multiplication*: $f \otimes g$ is a factor with domain $\text{dom}(f) \cup \text{dom}(g)$ and values $(f \otimes g)(\mathbf{v}) = f(\mathbf{v}) \cdot g(\mathbf{v})$.
- *Summation*: $\Sigma_{\mathbf{W}} f$, with $\mathbf{W} \subseteq \text{dom}(f)$, is a factor with domain $\mathbf{U} = \text{dom}(f) \setminus \mathbf{W}$ and values $(\Sigma_{\mathbf{W}} f)(\mathbf{u}) = \sum_{w \in \text{dom}(\mathbf{W})} f(\mathbf{u}, \mathbf{W}=w)$.

- *Indexing:* $f[\mathbf{W}=\mathbf{w}]$, with $\mathbf{W} \subseteq \text{dom}(f)$, is a factor with domain $\mathbf{U} = \text{dom}(f) \setminus \mathbf{W}$ and values $f[\mathbf{W}=\mathbf{w}](\mathbf{u}) = f(\mathbf{u}, \mathbf{W}=\mathbf{w})$.

In the literature, the notations $f^{\downarrow \mathbf{U}}$ for summation and $f|_{\mathbf{W}=\mathbf{w}}$ for indexing are often encountered. Furthermore, we use the notation $\mathbb{1}$ for the unit of factor multiplication; factor $\mathbb{1}$ has domain \emptyset and value 1.

Why not use conventional algebra?

Factor algebra is constructed in such a way that its *denotational* semantics are very close to conventional algebra expressions. For example, consider three factors f , g and h , with

$$\begin{aligned}\text{dom}(f) &= \{A\} & \text{dom}(A) &= \{a_1, a_2, a_3\} \\ \text{dom}(g) &= \{A, B\} & \text{dom}(B) &= \{b_1, b_2, b_3, b_4\} \\ \text{dom}(h) &= \{B, C\} & \text{dom}(C) &= \{c_1, c_2, c_3, c_4, c_5\}\end{aligned}$$

Then the result of the factor algebra expression

$$\Sigma_A(f \otimes \Sigma_B(g \otimes \Sigma_C h)) \quad (1)$$

is the same number as the conventional expression

$$\sum_{a \in \text{dom}(A)} f(a) \sum_{b \in \text{dom}(B)} g(a, b) \sum_{c \in \text{dom}(C)} h(b, c). \quad (2)$$

However, now consider the *operational* semantics of expr. (2). First, the outermost summation is expanded, replacing its formal variable a in the rest of the expression by concrete values a_1 , a_2 and a_3 :

$$\begin{aligned}& f(a_1) \sum_{b \in \text{dom}(B)} g(a_1, b) \sum_{c \in \text{dom}(C)} h(b, c) \\ & + f(a_2) \sum_{b \in \text{dom}(B)} g(a_2, b) \sum_{c \in \text{dom}(C)} h(b, c) \\ & + f(a_3) \sum_{b \in \text{dom}(B)} g(a_3, b) \sum_{c \in \text{dom}(C)} h(b, c)\end{aligned}$$

Then, each of the 3 b -summations are expanded into 4 terms; finally, each of the $3 \cdot 4$ c -summations are expanded into 5 terms. This leaves us with an expression with $3 + 3 \cdot 4$ multiplications and $3 \cdot 4 \cdot 5 - 1$ additions; running time is *exponential in the nesting depth*. Redundant calculations are made: the summation over C is performed for each of the 3 values for a , although its value does not depend on a .

On the other hand, expression (1) is evaluated from the inside out:

1. sum out C from h : $4 \cdot (5 - 1)$ additions
2. multiply with g : $3 \cdot 4$ multiplications
3. sum out B : $3 \cdot (4 - 1)$ additions
4. multiply with f : 3 multiplications
5. sum out A : $3 - 1$ additions

What matters here are not the precise numbers of additions or multiplications, but the fact that they do not depend exponentially on the expression's nesting depth, but rather on the *maximum domain size* (here: 2) of the intermediate factors produced in steps 1–5. Also, no redundant calculations are made this time. The difference between expr. (1) and (2) responsible for this is that in the latter, the concrete elements of $\text{dom}(A)$ are mentioned in the outermost summation, scoping over the complete expression and duplicating operations where such concrete elements are not needed. In expr. (1), all information about concrete domain elements comes from the inside, i.e. from factors f , g and h themselves.

The downside of evaluating factor algebra is that it requires more space. At each evaluation step, all the participating factors are completely present in memory. Thus, the required space is exponential in the maximum intermediate domain size as well.

To conclude: in factor algebra, rewriting the expression matters—expression (1) has a totally different performance than $\Sigma_{ABC}(f \otimes g \otimes h)$ —while in conventional algebra, expression (2) has the same nesting depth as $\sum_a \sum_b \sum_c f(a)g(a, b)h(b, c)$. With factor algebra as its basis, **symfer** enables the optimization of inference by rewriting expressions.

Factor algebra expressions as trees

Factor algebra expressions can be visualized as trees, like the example in Figure 2(a). The factors at the leaves of the tree are usually conditional probability tables that make up a Bayesian network; the branches are formed by \otimes , Σ and $[\mathbf{E}=\mathbf{e}]$ operations. The arrows in the tree illustrate the data flow when evaluating the expression; to evaluate a node, we first need to evaluate its children (in any order). The nodes in the tree can be annotated with properties like its factor domain and corresponding array size, to help understand the cost of evaluation.

The output of any of **symfer**'s symbolic inference algorithms is such a tree; in fact, Figure 2 is the output of variable elimination in the order $[A, B]$

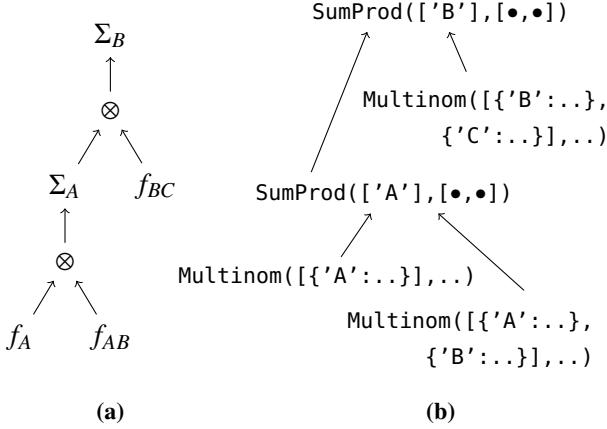


Figure 2: (a) Tree representation of factor algebra expression $\Sigma_B((\Sigma_A(f_A \otimes f_{AB})) \otimes f_{BC})$. (b) Python representation of this expression (also as tree; fill in the children at the \bullet positions to obtain the flat text).

(given that the factor domains are $\text{dom}(f_A) = \{A\}$, $\text{dom}(f_{AB}) = \{A, B\}$ and $\text{dom}(f_{BC}) = \{B, C\}$).

4 Representation of factors in Python

In the Python module, a factor expression is represented as a Python object. Its class corresponds to the top-level operation in the expression, and the object contains references to the arguments of this operation (themselves also objects representing factor expressions).

The *leaves* of the expression are factors representing conditional probability tables (or, in a Markov network, potentials), and are instances of the class `Multinom`. To construct such an instance, the user has to provide the factor's domain and a list of probabilities. The factor domain is a list with elements such as `{'Sprinkler': ['off', 'on']}`, i.e. a variable name and a variable domain (list of possible values). For example, two tables representing $P(\text{Weather})$ and $P(\text{Sprinkler}|\text{Weather})$ can be created as follows:

```
weather      = {'Weather': ['sunny', 'rainy']}
weather_cpd = Multinom([weather], [0.8, 0.2])
sprinkler_cpd = Multinom(
    {'Sprinkler': ['off', 'on']}, weather,
    [0.6, 0.4, 0.99, 0.01])
```

The tables are now stored as `weather_cpd` and `sprinkler_cpd`. We used an auxiliary Python variable `weather` to avoid repeating the definition of this domain. Not only does this save typing,

but the Python semantics also cause the domain to be *shared* in memory among `weather_cpd` and `sprinkler_cpd`; i.e. both contain a pointer to the same object.

Although it is possible to enter a probabilistic model in Python as shown above, we expect that most users will rather do this using a separate graphical editor. Therefore, `Multinom` instances can also automatically be read from Hugin files.

A *compound* factor algebra expression can be obtained constructing objects of classes `SumProd` and `Index`. The former represents a combination of the operations \otimes and Σ : the expression on the right hand side of the statement

```
sprinkler_marg = SumProd(['Weather'],
    [weather_cpd, sprinkler_cpd])
```

multiples the two tables, then sums out the variable 'Weather'. Of course, it is possible to represent only a product (by using an empty list of variables) or a summation (by using a singleton list of factors).

The result is assigned to Python variable `sprinkler_marg`. More accurately, the name `sprinkler_marg` in the local namespace is bound to the newly created `SumProd` object. This object just holds references to the two `Multinom` objects. Nothing is evaluated yet, but when this will eventually be the case, the `SumProd` object stipulates that the two factors are to be multiplied, and the 'Weather' variable is to be summed out. In this way, the `SumProd` object is a *symbolic representation* of the factor algebra expression. The correspondence is illustrated in Figure 2.

The Python objects that represent factors, such as `SumProd` and `Multinom` instances, are designed to be immutable: once they are constructed, their 'state' never changes. In this aspect, they are just like mathematical expressions. This design makes it possible to safely share subexpressions. For example, one could define:

```
joint= SumProd([], [weather_cpd, sprinkler_cpd])
sprinkler_marg = SumProd(['Weather'], joint)
weather_marg   = SumProd(['Sprinkler'], joint)
```

where the product of `weather_cpd` and `sprinkler_cpd` is shared among the two expressions that represent the marginals. This product `joint` represents the joint probability distribution over 'Weather' and 'Sprinkler', an immutable

```

1 def ve_order(facs,order):
2     for rv in order:
3         rv_sum = SumProd([rv],[f for f in facs if rv in f.domtypes])
4         other_facs = [f for f in facs if rv not in f.domtypes]
5         facs = [rv_sum] + other_facs
6     return I().product(*facs)

```

Figure 3: Variable elimination with predefined order. In line 6, the expression `I().product(*facs)` calls a convenience function that returns `SumProd([],fac)`, except when `fac` contains only one factor, in which case that factor is returned.

mathematical entity. This stands in contrast to designs where objects represent a current belief over a certain set of variables, as commonly found in software implementations of e.g. the junction tree algorithm.

Indexing (setting probabilistic variables to certain values) is done by creating an instance of class `Index`. For example, the part of the joint probability distribution where the weather is sunny, $P(\text{Sprinkler}, \text{Weather}=\text{sunny})$, is obtained using `Index({'Weather': 'sunny'}, joint)`. This is a factor over the single variable '`Sprinkler`'. Alternatively, we could have defined

```

evidence      = {'Weather': 'sunny'}
weather_sunny = Index(evidence, weather_cpd)
sprinkler_sunny= Index(evidence, sprinkler_cpd)
joint_sunny   = SumProd([], [weather_sunny, sprinkler_sunny])

```

which results in a different factor expression `joint_sunny` for the same distribution. The difference lies in the fact that in the former expression, the whole joint distribution is calculated (requiring, for example, the multiplication of $P(\text{Sprinkler}=\text{off}|\text{Weather}=\text{rainy})$ and $P(\text{Weather}=\text{rainy})$), whereas in `joint_sunny` only the (conditional) probabilities consistent with the evidence are considered.

5 Symbolic inference in symfer

In the previous section, we manually crafted a factor expression for $P(\text{Sprinkler}, \text{Weather}=\text{sunny})$ using the Python API; thus, we were performing inference for query `['Sprinkler']`, evidence `{'Weather': 'sunny'}` and a model consisting of `weather_cpd` and `sprinkler_cpd`. The symbolic inference algorithms in `symfer` perform this process automatically. They are defined using the same API. For example, the most simple inference algorithm,

variable elimination with a predefined elimination order, is given by the 6 lines in Figure 3. The function `ve_order` defined there is invoked as follows:

```
result = ve_order([weather_cpd,sprinkler_cpd],
                  ['Sprinkler', 'Weather'])
```

Note that the Python code is very close to mathematical pseudocode. For example, lines 3–5 would correspond to

$$\begin{aligned} \text{rv_sum} &\leftarrow \sum_{\text{rv}} \bigotimes \{ f \mid f \in \text{fac}, \text{rv} \in \text{dom}(f) \} \\ \text{other_fac} &\leftarrow \{ f \mid f \in \text{fac}, \text{rv} \notin \text{dom}(f) \} \\ \text{fac} &\leftarrow \{\text{rv_sum}\} \cup \text{other_fac} \end{aligned}$$

In `ve_order`, evidence is not explicitly considered. There are two approaches for including evidence. Either the user applies `Index` operations *before* invoking the function, i.e.

```
res_sunny = ve_order([weather_sunny,
                      sprinkler_sunny], ['Sprinkler'])
```

or the API function `indextree` is invoked on the function `result`:

```
res_sunny = indextree(evidence, result)
```

Here, the two approaches yield the same factor expression; however, when using an elimination heuristic such as `minweight`, they might result in a different elimination order.

The definition of variable elimination with a triangulation heuristic is not much more complex; see appendix A. Perhaps more surprisingly, junction tree propagation (in the Shenoy–Shafer variant (Shenoy and Shafer, 1988)) can be defined very concisely as well. The key insight is that the `SumProd` expression produced as a result of a variable elimination can function as a clique tree on which the algorithm is based.

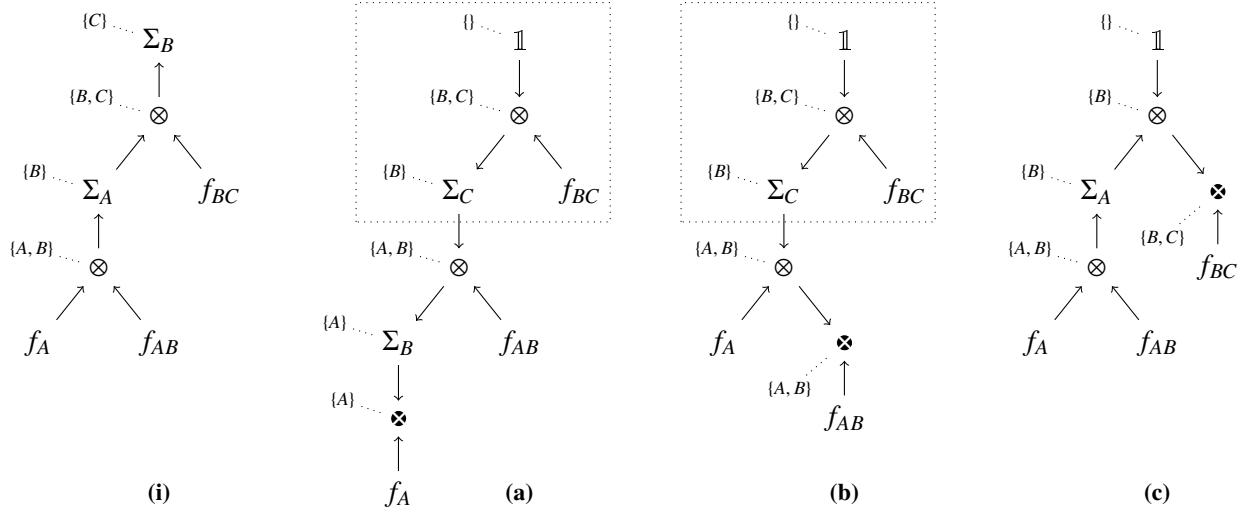


Figure 4: Example input (i) and output (a,b,c) of the junctiontree algorithm. Given a factor expression tree (such as the output of a variable elimination algorithm), junctiontree produces a new tree for each leaf (f_A, f_{AB}, f_{BC}) of the input tree. For clarity, we indicate the roots of these trees with \otimes here, and show the domain for each subtree using dotted lines (these are not part of the tree). For each output tree, a root node is created above the leaf, all the arrows on the path from this node to the original root are reversed, and sums are updated (or added). In memory, common subtrees such as in the dotted box (but also the f_A, f_{AB}, f_{BC} leaves) are shared among the output trees; see Figure 5. For the algorithm itself, see Appendix A.

In our variant, the algorithm creates a factor expression for each leaf of the original tree; this expression calculates the marginal probability distribution over the domain of this leaf. It does this by traversing the path from the original root (which is replaced by \mathbb{I} , or $\text{I}()$ in Python) down to the leaf in question, while creating a new tree in ‘reverse order’ from the subtrees not on this path; see Figure 4 for an example and Appendix A for the algorithm.

The fact that a junction tree can be created from the structure of a variable elimination procedure, and that junction tree propagation can be interpreted as N instances of variable elimination (one for each marginal) is well known (Shafer, 1996); the often cited advantage of the junction tree algorithm is that it theoretically runs in approximately $2\times$ the time of variable elimination instead of $N\times$. Traditionally, this is accomplished by explicitly storing intermediate results in nodes in the data structure. In our algorithm we achieve essentially the same results (using the same insight) by letting the output trees *share* their subtrees, as shown in Figure 5. As we explained before, this sharing happens quite transparently in Python: unless explicitly instructed, it

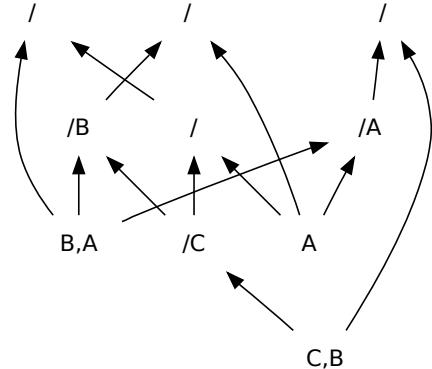


Figure 5: The output trees of the junction tree example in Figure 4, shown with explicit sharing of subtrees, as drawn by symfer and GraphViz. The top row lists, from left to right, the roots of (b), (a) and (c). A node starting with a slash represents a SumProd expression: it multiplies its children, then sums out the variables after the slash. The other nodes are Multinom instances. Multiplication with \mathbb{I} is simplified away.

never copies an object but always a pointer. Crucially, when the factor expressions are evaluated, the numeric stage is aware of this sharing, and evaluates the shared subexpressions only once. The serialization language YAML can represent this expression sharing, which is why it is used instead of JSON.

Although we have not implemented a specific algorithm yet, the principle of *conditioning*, see e.g. (Shachter et al., 1994) and (Darwiche, 2001), can also easily be expressed using factor algebra. For example, to condition on variable B in the expression in Figure 2 (suppose we have named it `veAB`), one could simply do

```
cb = [s.indextree({'B':b}, veAB)
      for b in ['b1','b2','b3']]
```

resulting in a list of 3 trees just like Figure 2(a), but without the Σ_B root node and with $[B=b_i]$ expressions at the leaves. The numeric stage would have to add the 3 resulting factors together; we will include support for this in a later version.

6 Inference in a telemedicine system

Apart from inference research, we are also putting `symfer` to use in a telemedicine system that we are developing (van der Heijden et al., 2011) for home use by COPD patients. COPD (chronic obstructive pulmonary disease) is a chronic lung disease, currently affecting some 210 million people worldwide, with considerable health care related costs. The progression of the disease can be slowed by rapid intervention when an exacerbation (acute worsening of the symptoms) is detected. Our system, which consists of a smartphone and two small physiological sensors, enables a patient to perform simple measurements and fill in a questionnaire about the symptoms in a home (or mobile) setting on a daily basis.

Using a probabilistic disease model, the system then determines whether the chance of an exacerbation is high enough to warrant an intervention. It sends its findings to an authorised physician, but can also directly advise the patient. For the system to work without network communication, the probabilistic inference is performed on the Android smartphone itself.

Until recently, we used an inference library on the phone that required us to define the

`model+query+evidence` and run a complete inference algorithm every time the application was started. Currently, we use `symfer`, in a setup where we provide the symbolic stage with `model` and `query` variable and run it only once (on a separate machine). This is possible because in our application, the query does not change. On the smartphone, only the numeric stage is performed. Every time the application is run, the sensors and questionnaire provide new evidence; as we explained in section 2, the numeric stage can insert this evidence in the factor expression just before it evaluates it.

As it has the same structure but starts with smaller arrays at the leaves, the expression with evidence is guaranteed not to require more memory than the original expression without evidence. Thus, if the smartphone has enough memory to run the original expression (which we need to test only once), it can also run the inference with any combination of evidence. This would not have been the case if we ran a variable elimination heuristic each time, as this can change the structure of the expression.

7 Conclusion and prospects

What distinguishes `symfer` from other inference software is its thorough separation of inference algorithms into a symbolic and numeric stage. We have shown that at the former stage, widely taught algorithms such as variable elimination, junction tree propagation and conditioning can be formulated in a unified manner. This is achieved through the central role of factor algebra, combined with the sharing semantics inherent in Python.

Furthermore, the Python language allowed us to express the algorithms in a notation that is almost as clear and concise as pseudocode. We expect this to facilitate teaching and explorative research. Here, the interactive Python interpreter (Pérez and Granger, 2007) will also help, in the same way as MATLAB does.

Apart from these advantages, we have found the separation of the two stages useful for deploying part of the software on the Android platform, where it is part of a telemedicine application. The reason for this is that the numeric stage is quite small and easily portable. This also means that it can be *optimized* separately from the rest of the software

(for example, by data management researchers, not needing any knowledge of probabilistic models).

We are currently working on support for recursive conditioning, factor indexing (Evers and Lucas, 2011), and efficient handling of deterministic patterns such as Noisy-OR; it is no problem to express these in the factor algebra framework. A somewhat larger challenge will be to include continuous variables, but we expect this to be possible as well.

We make `symfer` available as open source (BSD-licensed) software, except for the Android numeric evaluator, which we plan to publish with a license for free academic use.

References

- Adnan Darwiche. 2001. Recursive conditioning. *Artif. Intell.*, 126(1-2):5–41.
- Sander Evers and Peter J. F. Lucas. 2011. Marginalization without summation: Exploiting determinism in factor algebra. In Weiru Liu, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 11th European Conference, ECSQARU 2011. Proceedings*, volume 6717 of *Lecture Notes in Computer Science*, pages 251–262. Springer.
- R.E. Kalman. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.
- D. Koller and N. Friedman. 2009. *Probabilistic graphical models: Principles and techniques*. MIT Press.
- Fernando Pérez and Brian E. Granger. 2007. IPython: a System for Interactive Scientific Computing. *Comput. Sci. Eng.*, 9(3):21–29, May.
- Ross D. Shachter, Bruce D’Ambrosio, and Brendan Del Favero. 1990. Symbolic probabilistic inference in belief networks. In Howard E. Shrobe, Thomas G. Dietterich, and William R. Swartout, editors, *AAAI*, pages 126–131. AAAI Press / The MIT Press.
- Ross D. Shachter, Stig K. Andersen, and Peter Szolovits. 1994. Global conditioning for probabilistic inference in belief networks. In Ramon López de Mántaras and David Poole, editors, *UAI ’94: Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 514–522. Morgan Kaufmann.
- Glenn Shafer. 1996. *Probabilistic expert systems*. Society for Industrial and Applied Mathematics.
- Prakash P. Shenoy and Glenn Shafer. 1988. Axioms for probability and belief-function propagation. In Ross D. Shachter, Tod S. Levitt, Laveen N. Kanal, and John F. Lemmer, editors, *UAI ’88: Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*, pages 169–198. North-Holland.
- Maarten van der Heijden, Bas Lijnse, Peter J. F. Lucas, Yvonne F. Heijdra, and Tjard R. J. Schermer. 2011. Managing COPD exacerbations with telemedicine. In Mor Peleg, Nada Lavrac, and Carlo Combi, editors, *13th Conference on Artificial Intelligence in Medicine, AIME 2011. Proceedings*, volume 6747 of *Lecture Notes in Computer Science*, pages 169–178. Springer.
- Nevin Lianwen Zhang and David Poole. 1996. Exploiting causal independence in bayesian network inference. *J. Artif. Intell. Res. (JAIR)*, 5:301–328.
- ```

1 def ve_minweight(facs,query):
2 remaining = set().union(
3 *[set(fac.domlist) for fac in facs])
4 remaining -= set(query)
5 while remaining:
6 cand = None,None,float('inf')
7 for rv in remaining:
8 rv_facs = [fac for fac in facs
9 if rv in fac.domtypes]
10 rv_prod = I().product(*rv_facs)
11 rv_weight = len(rv_prod)
12 if rv_weight < cand[2]:
13 cand = rv,rv_prod,rv_weight
14 rv,rv_prod,_ = cand
15 remaining.remove(rv)
16 rv_sum = rv_prod.sumout([rv])
17 other_facs = [fac for fac in facs
18 if rv not in fac.domtypes]
19 facs = [rv_sum] + other_facs
20 return I().product(*facss)

1 def junctiontree(tree,upfac=I()):
2 if isinstance(tree,SumProd):
3 out = []
4 for f in tree.fac:
5 prod_others = upfac.product(*[other for other in tree.fac if other is not f])
6 facres = junctiontree(f,prod_others.sumto(f.domlist))
7 out.extend(facres)
8 return out
9 elif isinstance(tree,Multinom):
10 return [upfac.product(tree)]

```

# Meta-prediction of semi-naive Bayesian network classifiers based on dataset complexity characterization

M. Julia Flores, José A. Gámez, Ana M. Martínez

Universidad de Castilla-La Mancha, Spain

{julia.flores,jose.gamez,anamaria.martinez}@uclm.es

## Abstract

Ever since naive Bayes was proposed, many have been the attempts to try to alleviate its naive assumption to obtain better accuracy records, without further increasing its complexity. In this line we can find a group of Bayesian network classifiers that either do not perform structural search or it is very simple, known as the family of semi-naive Bayesian network classifiers (BNCs). Given a particular dataset, it would be desirable, based on the characteristics presented, to find out which semi-naive BNCs obtains the best possible expected prediction, since estimations based solely on expected accuracy on training can be misleading. In this paper we propose an automatic procedure to carry out this meta-prediction process, based on the values of several data complexity measures for supervised classification. We resort to multi-label classification to test this procedure, obtaining promising results.

## 1 Introduction

Learning the structure of a network can take a long time and effort, especially for datasets of high dimensionality. That is why it is often convenient to consider a partially or totally pre-fixed structure from which the conditional probability distributions are learnt. The most simple of these structures is the one used by naive Bayes (NB), that assumes all the attributes are independent given the class. In spite of its naive assumption, it performs surprisingly well in some domains. Many techniques improve the accuracy of NB by alleviating the attribute interdependence problem, such as Averaged One-Dependence Estimators (AODE), Tree augmented naive Bayes (TAN) or  $k$ -dependence Bayesian classifier (KDB). All these techniques, known as semi-naive BNCs, do not perform structural search or this search is very simple (Flores et al., 2012).

It is still unclear, at the moment of writing, which of these classifiers should be used for a particular dataset. There have been several studies oriented to compare classifiers of different nature. In Ho and Basu (2002), a selection

of several measures for characterizing the complexity of classification problems is presented, along with an empirical study on the distribution of real world problems compared to random noise, indicating that it is possible to find learnable structures with the geometrical measures presented. These measures indicate the overlap of individual feature values; the separability of classes; and geometry, topology and density of manifolds. This group of measures encounters its natural definition in the two-class domain. Nevertheless, attempts to generalize some of these measures to the multi-class domain can be found in Mollineda et al. (2005) and more recently in Orriols-Puig et al. (2010).

Numerous studies have followed that try to obtain the domains of competence for one or more particular classifiers, by studying error rate patterns with respect to individual or combination of complexity measures (CMs), usually bivariate combinations. Some of these works are Bernadó-Mansilla and Ho (2004) for 1-nearest-neighbour (1NN), linear classification, decision trees and decision forests; Sánchez et al. (2007) for kNN; and more

recently, Luengo and Herrera (2010) and Luengo and Herrera (2012) for fuzzy rule based classification systems, or artificial neural networks and support vector machines.

Note that none of these studies focuses on the family of semi-naive BNCS. As a different and more practical approach, we propose an automatic mechanism to select the most promising semi-naive BNC for a particular dataset, based on the values of some of the CMs. An interesting work in relation to this topic is presented in Hernández-Reyes et al. (2005), where an automatic classifier selection based on data complexity measures is proposed. Their method describes problems with complexity measures and labels them with the classifier that gets the best accuracy among a set of five classifiers: kNN, NB, linear regression, RBFNetwork and J48. We argue that simply selecting the classifier that obtains the highest accuracy is inaccurate, since several classifiers can be equally good for a given problem. We present an alternative procedure based on *partial* multi-label classification, where the term partial refers to the use of the multi-label paradigm for the training phase exclusively, and empirically test this procedure.

The paper is divided as follows: Section 2 introduces the semi-naive BNC considered. In Section 3 the existing data complexity measures for supervised classification are presented. Section 4 introduces the multi-label classification problem. Section 5 outlines our proposal for automatic meta-prediction of the semi-naive BNCs. The empirical results obtained are presented in Section 6. Section 7 includes the main conclusions and future work.

## 2 Semi-naive Bayesian network classifiers

Apart from NB, the following semi-naive BNCs will be considered:

**AODE:** (Webb et al., 2005) is an ensemble of  $n$  models (where  $n$  is the number of attributes) in which every attribute depends on the class and another shared attribute (at model  $i$ ,  $A_i$ ), designated as superparent. AODE is an attractive alternative to other approaches that aim to

improve NB maintaining its efficiency, as it provides competitive error rates with an efficient profile (Zheng and Webb, 2005).

**HODE** (Flores et al., 2009) estimates a new hidden variable using the EM algorithm (Dempster et al., 1977), whose main objective is to model the meaningful dependences between each attribute and the rest of the attributes that AODE takes into account. HODE can be considered an attractive alternative to AODE, especially in datasets where the number of attributes or number of values per attributes is very large.

**TAN:** (Friedman et al., 1997) learns a maximum weighted spanning tree based on the conditional mutual information between two attributes given the class label. Then, the arcs in the tree are oriented by choosing a root and the model is completed by adding a link from the class to each attribute.

**KDB:** Sahami (1996) introduced the notion of  $k$ -dependence estimators, from which the probability of each attribute value is conditioned by the class and, at most,  $k$  other attributes. Throughout the KDB algorithm it is possible to construct classifiers across the whole spectrum, from the NB structure to the full BN structure, by varying the value of  $k$ , i.e. the maximum number of parents that every attribute can have.

## 3 Data complexity measures

The complexity measures originally proposed in Ho and Basu (2002) have already shown their power for characterizing classifiers of different nature, although mainly on continuous datasets. Some of the CMs have been originally defined for numeric values. Since the natural domain of the Bayesian network classifiers is with discrete variables, nominal attributes will be mapped into integer numbers for the calculations of these measures, assuming though a non-existent order between the labels.

These complexity measures are summarized in Table 1, where as a novelty, a new column is added to indicate the tendency that a particular measure may follow according to its definition,

| Type                                                     | Identifier | Name/Description                                                | Simpler if... |
|----------------------------------------------------------|------------|-----------------------------------------------------------------|---------------|
| Overlaps in the feature values from different classes    | F1         | Maximum Fisher's discriminant ratio                             | +             |
|                                                          | F1v        | Directional-vector maximum Fisher's discriminant ratio          | +             |
|                                                          | F2         | Overlap of the per-class bounding boxes                         | -             |
|                                                          | F3         | Maximum (individual) feature efficiency                         | +             |
|                                                          | F4         | Collective feature efficiency                                   | +             |
| Measures of class separability                           | L1         | Minimized sum of the error distance of a linear classifier      | -             |
|                                                          | L2         | Training error of a linear classifier                           | -             |
|                                                          | N1         | Fraction of points on the class boundary                        | -             |
|                                                          | N2         | Ratio of average intra/inter class nearest neighbor distance    | -             |
|                                                          | N3         | Leave-one-out error rate of the one-nearest neighbor classifier | -             |
| Measures of geometry, topology, and density of manifolds | L3         | Nonlinearity of a linear classifier                             | -             |
|                                                          | N4         | Nonlinearity of the one-nearest neighbor classifier             | -             |
|                                                          | T1         | Fraction of maximum covering spheres                            | -             |
|                                                          | T2         | Average number of points per dimension                          | +             |

Table 1: Summary of CMs for supervised classification.

in order to reflect more simplicity on the problem it applies. Note that the symbol + indicates more simplicity as the value of the corresponding complexity measure increases, and - as it decreases.

#### 4 Multi-label classification

Multi-label classification is a type of classification problem where multiple class labels can be assigned to each example<sup>1</sup>.

A comprehensive overview on multi-label classification can be found in Tsoumakas and Katakis (2007). Numerous multi-label classification techniques are gathered, mainly divided into problem transformation and algorithm adaptation methods. The first group of methods transform the learning task into one or more single-label classification tasks, whereas the second group extend specific learning algorithms in order to handle multi-label data directly.

We are using two classifiers that belong to the category of problem transformation methods: NB using the binary relevance (NB-BR) transformation method; and the RAndom  $k$ -labELsets method (RAkEL). BR is one of the simplest and most popular transformation methods; where  $L$  single-label classifiers ( $L$  being the number of class labels) are learned from the  $L$  class-binary datasets created. RAkEL uses an ensemble of label powerset (LP) trans-

<sup>1</sup>Also known as multi-dimensional classification (Bielza et al., 2011). Note, in any case, the difference with a multi-class problem, that simply refers to the existence of one class with more than two labels.

formation methods. LP considers each unique set of labels that exists in a multi-label training set as one of the classes of a new single-label classifier.

#### 5 Meta-classification of semi-naive BNCs

The main objective is that, given a particular dataset to be classified, it is possible to predict which semi-naive BNC is more likely to provide the most accurate predictions. For this purpose, it is necessary to create what we call a *training meta-dataset*: where every instance represents a single dataset, for which the predictive attributes correspond, in principle, to the 14 complexity measures in Table 1. This is in concordance with the method proposed in Hernández-Reyes et al. (2005). One of the main differences between their approach and ours is that they consider a single class label dataset, assigning to every instance the classifier with the lowest error for the dataset that represents that instance.

In our case, 5 semi-naive BNCs for discrete attributes are considered, in particular: NB, AODE, HODE, TAN and KDB<sup>2</sup>. We believe that the selection of a single classifier based directly on the lowest error value can be too arbitrary. Alternatively, we propose to carry out statistical tests on the classifiers' results for each problem, in order to keep the best classifier and also those whose error rates are not significantly

<sup>2</sup>KDB with  $k = 3$  has been selected for these experiments in order to gain some variety among the classifiers considered.

Table 2: Main characteristics of the 26 numeric datasets: number of predictive variables ( $n$ ), number of classes ( $c$ ) and number of instances ( $m$ ).

| Id | Datasets        | $n$ | $c$ | $m$   | Id | Datasets      | $n$ | $c$ | $m$   |
|----|-----------------|-----|-----|-------|----|---------------|-----|-----|-------|
| 1  | balance-scale   | 4   | 3   | 625   | 14 | mfeat-fourier | 76  | 10  | 2000  |
| 2  | breast-w        | 9   | 2   | 699   | 15 | mfeat-karh    | 64  | 10  | 2000  |
| 3  | diabetes        | 8   | 2   | 768   | 16 | mfeat-morph   | 6   | 10  | 2000  |
| 4  | ecoli           | 7   | 8   | 336   | 17 | mfeat-zernike | 47  | 10  | 2000  |
| 5  | glass           | 9   | 7   | 214   | 18 | optdigits     | 64  | 9   | 5620  |
| 6  | hayes-roth      | 4   | 4   | 160   | 19 | page-blocks   | 10  | 5   | 5473  |
| 7  | heart-statlog   | 13  | 2   | 270   | 20 | pendigits     | 16  | 9   | 10992 |
| 8  | ionosphere      | 34  | 2   | 351   | 21 | segment       | 19  | 7   | 2310  |
| 9  | iris            | 4   | 3   | 150   | 22 | sonar         | 60  | 2   | 208   |
| 10 | kdd-JapanV      | 14  | 9   | 9961  | 23 | spambase      | 57  | 2   | 4601  |
| 11 | letter          | 16  | 26  | 20000 | 24 | vehicle       | 18  | 4   | 946   |
| 12 | liver-disorders | 6   | 2   | 345   | 25 | waveform-5000 | 40  | 3   | 5000  |
| 13 | mfeat-factors   | 216 | 10  | 2000  | 26 | wine          | 13  | 3   | 178   |

worse. Given that the considered classifiers belong to the same family, it is reasonable to expect small differences.

This then requires to resort to multi-label classification in order to handle the existence of multiple class labels.  $5\times 2cv$  is used for the evaluation process, and the  $5\times 2cv$  F Test defined by Alpaydin (1999) has been used to select the semi-naive BNCs for each dataset (unilateral comparisons). The level of significance has been fixed at 5% ( $\alpha = 0.05$ ).

In order to construct the meta-dataset with the complexity measure values from different datasets, we select the group of 26 numeric datasets in Table 2. Since most of the CMs are only defined to deal with datasets with two class labels, we have created several binary datasets from each dataset with more than 2 class labels, specifically, as many as the total number of class labels per dataset (by following the strategy known as one-against-all). Hence, there is a total of 157 datasets with two class labels. Additionally, we discretize them applying unsupervised equal frequency discretization with 5 bins. Motivated by the work in Flores et al. (2011) we believe that the choice of the discretization technique is not decisive in this context.

A small sample of the resulting training meta-dataset is shown in Table 3. Every example corresponds to the result of the 14 complexity measures for a specific dataset, whereas the class

|                                            |                           |
|--------------------------------------------|---------------------------|
| <b>Examples:</b> 157                       | <b>Labels:</b> 5 (binary) |
| <b>Predictive attributes:</b> 14 (numeric) |                           |
| <b>Distinct Labelsets:</b> 24              |                           |
| <b>Cardinality:</b> 2.52                   | <b>Density:</b> 0.50      |
| <b>*Percentage of examples with label:</b> |                           |
| 1(NB): 19.74%                              | 4(TAN): 52.23%            |
| 2(AODE): 56.59%                            | 5(KDB3): 61.78%           |
| 3(HODE): 61.15%                            |                           |
| <b>*Examples of cardinality:</b>           |                           |
| 0: 0                                       | 3: 43                     |
| 1: 39                                      | 4: 19                     |
| 2: 43                                      | 5: 13                     |

Table 4: Statistics of the meta-dataset created.

labels are binary and correspond to the 5 following semi-naive BNCs: NB, AODE, HODE, TAN and KDB3: a bit equal to 1 for the  $j^{th}$  class label on the  $i^{th}$  instance indicates that the classifier on  $j$ , either obtain the best error rate for the dataset on the  $i^{th}$  position or it is not significantly worse than the classifier that does.

The statistics of the resulting meta-dataset are summarized in Table 4. The upper part of the table (above the horizontal line) displays general information: such as the number of examples, attributes or class labels; whereas the lower part includes more specific information related to the class labels. The number of distinct labelsets indicates the binary combinations out of the  $2^5$  possible. The value for cardinality is calculated as the ratio of positive bits (those equal to 1) in the labels over the total number of instances. The density is just the division of the cardinality between the number of labels. These values indicate that, on average, every example can be optimally classified by at least 2 semi-naive BNCs. Two figures to be highlighted here are: the number of examples of cardinality equal to 1, i.e., those that only have one “best” classifier, which is 39; and the number of trivial examples (cardinality equal to 5), i.e., those for which any classifier would be equally eligible, which is 13.

For our experiments, we select the following two approaches to carry out the meta-classification task:

- **NB-BR:** (Tsoumakas and Katakis, 2007)  
In our case, we transform the original dataset into 5 datasets with binomial class,

Table 3: Sample of the meta-dataset created to predict the best semi-naive BNC based on data complexity measures.

| dataset    | F1    | F1v    | F2    | F3    | F4    | L1    | L2    | L3    | N1    | N2    | N3    | N4    | T1 | T2   | NB | AODE | HODE | TAN | KDB3 |
|------------|-------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|----|------|----|------|------|-----|------|
| a1iris.2c0 | 0.474 | 16.455 | 0.000 | 0.807 | 1.000 | 0.333 | 0.007 | 0.000 | 0.067 | 0.209 | 0.013 | 0.263 | 1  | 37.5 | 0  | 1    | 0    | 0   | 0    |
| a1iris.2c1 | 0.458 | 2.374  | 0.250 | 0.433 | 0.573 | 0.658 | 0.333 | 0.500 | 0.200 | 0.264 | 0.093 | 0.250 | 1  | 37.5 | 0  | 0    | 1    | 0   | 0    |
| a1iris.2c2 | 0.475 | 10.186 | 0.062 | 0.627 | 0.760 | 0.461 | 0.053 | 0.017 | 0.160 | 0.244 | 0.093 | 0.290 | 1  | 37.5 | 1  | 1    | 1    | 0   | 1    |
| :          | :     | :      | :     | :     | :     | :     | :     | :     | :     | :     | :     | :     |    | :    | :  | :    | :    | :   |      |

each one containing the corresponding column (NB,AODE,...) as class. For the classification of a new instance, the original definition of BR would output the union of the labels that are positively predicted by the 5 classifiers. In our experiments, only the most likely label is considered.

- **RakEL:** (Tsoumakas and Vlahavas, 2007), is a random  $k$ -labelset method that constructs an ensemble of LP classifiers (of type C4.5 in our experiments), where each LP is trained using a different small random subset of size  $k$  of the set labels. A ranking of the labels is produced by averaging the zero-one predictions of each model per considered label. Thresholding is used to produce a bipartition as well. Only the first label in the ranking will be considered in our case.

However, all of these strategies consider a multi-label prediction phase as well. For our purposes, even though we are training a classifier based on a multi-label paradigm, we select only the best classifier to predict with. This restriction requires to redefine the way in which the evaluation is performed, so that a specific prediction,  $Z_e$  for the example  $(e, Y_e)$ , is considered successful if the label predicted is among those included in  $Y_e$ , where both  $Z_e$  and  $Y_e$  are binary vectors of length  $L$ ,  $L$  being the number of semi-naive BNCs considered, i.e., the number of labels. However, given that we have modified the output such that the number of positive values in  $Z_e$  is only one, we can use the original definition of **example-based precision** (Tsoumakas et al., 2010) as evaluation measure:

$$Precision = \frac{1}{m} \sum_{i=1}^m \frac{|Y_i \cap Z_i|}{|Z_i|}, \quad (1)$$

where the operator  $|.|$  indicates the cardinality of the positive bits. Consider, for example, an output  $Y_e = \{0, 1, 1, 0, 0\}$ , which indicates that both AODE and HODE provide the best results for a particular dataset  $e$ , i.e., one of them has the highest absolute value and the other is not significantly worse. Then, if the meta-classifier, let say NB with BR (NB-BR), provides the output  $Z_e = \{0, 1, 0, 0, 0\}$ , this example would contribute to the summation as 1. If the output provided by RakEL were  $Z_e = \{0, 0, 0, 0, 1\}$  instead, the contribution would be equal to 0.

Note that since the average number of “valid” labels for every instance is equal to 2.5, our classification problem can be considered to be of equivalent difficulty to a binary class problem, since there is a 50% of probability to be accurate when classifying.

Additionally, it may not be necessary to use all the complexity measures as predictive attributes, as some of them can be redundant, irrelevant and maybe the “intrinsic” dimensionality may be smaller than the total number of measures considered. To that aim, we find a large amount of literature for feature selection (FS) (Guyon and Elisseeff, 2003). Note that FS is not required here for dimensionality reduction with efficiency purposes, but it could be beneficial to remove measures that are too similar in the meta-dataset, and hence, redundant.

The whole process is outlined in Figure 1. The left-hand side displays the three steps involved in the meta-dataset formation, which entails the most time consuming part of the process. The steps required when a new dataset faces a classification process, is included in the

dashed line. Given a new dataset, the values for the CMs considered in the meta-classification process will be calculated (not necessarily all of them, as shown in Section 6). From these values the meta-classifier selected will return the best semi-naive BNC.

## 6 Experimental methodology and results

We have resorted to a Java library for multi-label learning, called Mulan (Tsoumakas et al., 2011), in order to handle the multiple labels. The two meta-classifiers selected to work with the meta-dataset created are NB with BR and RAKEL, described above. The calculations of the different measures have been obtained with the data complexity library in C++ (Orriols-Puig et al., 2010).

10-fold cross-validation has been used for evaluation. The selection of these two multi-label classifiers have been motivated by the results obtained with the different algorithms provided by Mulan. Other paradigms have been tested, such as a lazy learning approach based on kNN, ML-KNN (Zhang and Zhou, 2007); and transformation methods, such as classifier chains (Read et al., 2011) with different base classifiers. Even though this study is not an exhaustive one, since it does not cover all the multi-label classifiers in the existing literature, for instance (Bielza et al., 2011), we believe that it is sufficient for our purposes.

In Table 5, different results in terms of example-based precision are shown. The alternatives tested are as follows:

- The first column, **Data**, indicates whether the data considered are directly the value of the measures for the different datasets (Original) or the data have been transformed through principal component analysis techniques (PCA). Dimensionality reduction is accomplished by choosing enough eigenvectors to account for a percentage of the variance in the original data, which has been set to 95%<sup>3</sup> (PC

<sup>3</sup>Default value in WEKA.

space). Furthermore, the PC space data have been transformed back to the original space eliminating some of the worst eigenvectors, with the aim of filtering attribute noise (PC space transformed back to original space).

- Feature selection through clustering techniques has also been carried out in some cases, as indicated in the second column, **Clustering+FS**. More specifically, a k-means clustering algorithm (using Euclidean distance) is performed in the transposed dataset with<sup>4</sup>  $k = 10$ . The output indicates the following clusters: ( $L_1, N_2$ ), ( $L_2, L_3, N_3$ ), ( $F_1, N_1$ ) and the rest of the measures in isolation. In order to select which measures to keep from each cluster we use PCA techniques again. The procedure is as follows: data are transformed through the PC space and back to the original space. As only the best PCs are retained, by setting the variance covered equal to 95%, we will obtain a dataset in the original space but with less attribute noise as above. Hence, the ranking obtained by this method is:

$$F_4, L_2, L_1, F_{1v}, F_1, F_3, F_2, N_4, T_2, T_1, N_1, L_3, N_3, N_2$$

We maintain then:  $L_1$  from cluster ( $L_1, N_2$ ),  $L_2$  from ( $L_2, L_3, N_3$ ) and  $F_1$  from ( $F_1, N_1$ ), along with the rest of the measures. And we will discard  $N_2$ ,  $L_3$ ,  $N_3$  and  $N_1$ , which happen to be the last four attributes given by PCA.

- Two multi-label classifiers (BR-NB and RAKEL) are directly applied or after performing FS as explained above. Indicated in the third column, **Meta-Classifier**.

The results on the last column in Table 5, show a variety of accuracy values ranging from

<sup>4</sup>Note that in this case, the purpose of feature selection is mainly carried out in order to remove possible noisy features, that is why we consider appropriate (although arbitrarily) not to remove more than 4 predictive attributes.

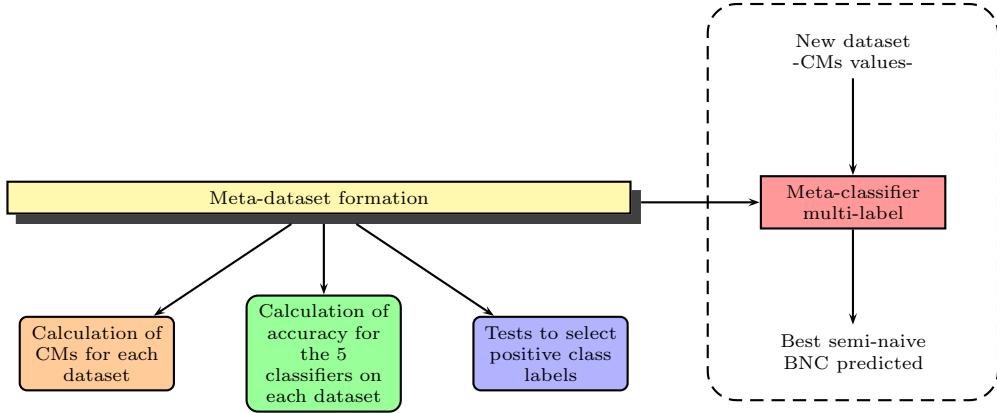


Figure 1: Schema of the meta-classification process.

Table 5: Expected example-based precision for meta-classifier selection.

| Data                                        | Clustering+FS | Meta-Classifier | Precision ± Stand. dev. |
|---------------------------------------------|---------------|-----------------|-------------------------|
| Original                                    |               | BR-NB           | 84.79±7.40              |
|                                             |               | RAkEL           | 86.75±7.59              |
|                                             | K-means+PCA   | BR-NB           | 86.08±5.32              |
|                                             | K-means+PCA   | RAkEL           | 86.04±7.30              |
| PC space                                    |               | BR-NB           | 85.46±10.4              |
|                                             |               | RAkEL           | 77.67±8.61              |
| PC space transformed back to original space |               | BR-NB           | 86.08±6.63              |
|                                             |               | RAkEL           | 86.71±5.8               |
|                                             | K-means+PCA   | BR-NB           | 86.08±6.01              |
|                                             |               | RAkEL           | <b>87.38±7.81</b>       |

77.7% to 87.4% depending on the data considered, the use or not of pre-processing techniques for FS and the multi-label classifier applied. It is obvious that the options and combinations here to test with are massive, and it is not our aim to perform an exhaustive study. The main purpose of this small comparison is to give an idea of the predictive power of the model.

All in all, the results seem to be encouraging, since in the best case, they offer a precision estimated in 87.38% of predicting correctly one of the best semi-naive BNCs, based on the complexity measures of a particular dataset with discrete attributes.

## 7 Conclusions

This paper presents an automatic procedure to advise on the best semi-naive BNC to use for classification. A meta-data set is created with the values of different CMs as predictive attributes. Then, a multi-label classifier is trained and afterwards used to predict a single best

semi-naive BNC, given the values of a subgroup of CMs of the target dataset.

This procedure has been tested to select among 5 semi-naive BNCs. A promising estimated predictive accuracy of 87.38% has been obtained with the RAkEL multi-label classifier, and after preprocessing the meta-dataset created.

As future work, the test bed of considered datasets can be extended incorporating the datasets from the Landscape contest (Macià et al., 2010), that has been created to cover a wider range of the complexity measurement space.

## Acknowledgments

This work has been financially supported by the FPU grant with reference number AP2007-02736, and it has also been partially funded by FEDER funds and the Spanish Government (MICINN) through project TIN2010-20900-C04-03.

## References

- E. Alpaydin. 1999. Combined  $5 \times 2$  cv F test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892.
- E. Bernadó-Mansilla and T. K. Ho. 2004. On classifier domains of competence. In *Proc. of ICPR'04, Volume 1 - Volume 01*, pages 136–139.
- C. Bielza, G. Li, and P. Larrañaga. 2011. Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727, September.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39:1–38.
- M. J. Flores, J. A. Gámez, A. M. Martínez, and J. M. Puerta. 2009. HODE: Hidden One-Dependence Estimator. In *Proc. of ECSQARU '09*, pages 481–492. Springer-Verlag.
- M. J. Flores, J. A. Gámez, A. M. Martínez, and J. M. Puerta. 2011. Handling numeric attributes when comparing Bayesian network classifiers: does the discretization method matter? *Applied Intelligence*, 34:372–385, June.
- M. J. Flores, Gámez J. A., and Martínez A. M. 2012. Intelligent data analysis for real-life applications: Theory and practice. chapter Supervised classification with Bayesian networks: A review on models and applications, pages 72–102. IGI Global.
- N. Friedman, D. Geiger, and M. Goldszmidt. 1997. Bayesian network classifiers. *Machine Learning*, 29:131–163.
- I. Guyon and A. Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March.
- E. Hernández-Reyes, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad. 2005. Classifier selection based on data complexity measures. In *Proc. of CIARP'05*, pages 586–592. Springer-Verlag.
- T. K. Ho and M. Basu. 2002. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:289–300.
- J. Luengo and F. Herrera. 2010. Domains of competence of fuzzy rule based classification systems with data complexity measures: A case of study using a fuzzy hybrid genetic based machine learning method. *Fuzzy Sets and Systems*, 161:3–19.
- J. Luengo and F. Herrera. 2012. Shared domains of competence of approximate learning models using measures of separability of classes. *Information Sciences*, 185(1):43–65, February.
- N. Macià, T. K. Ho, A. Orriols-Puig, and E. Bernadó-Mansilla. 2010. The landscape contest at ICPR 2010. In *Proc. of ICPR'10*, pages 29–45. Springer-Verlag.
- R. A. Mollineda, J. S. Sánchez, and J. M. Sotoca. 2005. Data characterization for effective prototype selection. In *Proc. of IbPRIA*, pages 27–34. Springer.
- A. Orriols-Puig, N. Macià, and T. K. Ho. 2010. Documentation for the data complexity library in C++. Technical report, La Salle - Universitat Ramon Llull.
- Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359.
- M. Sahami. 1996. Learning limited dependence Bayesian classifiers. In *Proc. of KDD'96*, pages 335–338.
- J. S. Sánchez, R. A. Mollineda, and J. M. Sotoca. 2007. An analysis of how training data complexity affects the nearest neighbor classifiers. *Pattern Analysis and Applications*, 10:189–201, July.
- G. Tsoumakas and I. Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13.
- G. Tsoumakas and I. Vlahavas. 2007. Random k-labelsets: An ensemble method for multilabel classification. In *Proc. of ECML '07*, pages 406–417. Springer-Verlag.
- G. Tsoumakas, I. Katakis, and I. Vlahavas. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685.
- G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. 2011. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414.
- G. I. Webb, J. R. Boughton, and Z. Wang. 2005. Not so naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, 58(1):5–24.
- M.-L. Zhang and Z.-H. Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.
- F. Zheng and G. I. Webb. 2005. A comparative study of semi-naïve Bayes methods in classification learning. In *Proc. of AusDM*, pages 141–156.

# Gaussian Process Regression with Censored Data Using Expectation Propagation

Perry Groot, Peter Lucas  
Radboud University Nijmegen, the Netherlands  
[{perry,peterl}cs.ru.nl](mailto:{perry,peterl}cs.ru.nl)

## Abstract

Censoring is a typical problem of data gathering and recording. Specialized techniques are needed to deal with censored (regression) data. Gaussian processes are Bayesian non-parametric models that provide state-of-the-art performance in regression tasks. In this paper we propose an extension of Gaussian process regression models to data in which some observations are subject to censoring. Since the model is not analytically tractable we use Expectation propagation to perform approximate inference on it.

## 1 Introduction

The method of data gathering and recording is a central issue in data analysis. A typical practical problem is censoring, which occurs when the value of a measurement or observation is only partially known. For example, censoring occurs when monitoring university freshmen in order to predict the drop out after the first semester. Another example of censoring occurs when a value occurs outside the range of a measuring instrument like a scale with a fixed limit. Censoring is also typical in clinical studies when individuals may withdraw from the study prematurely. Censored regression models were first suggested in Econometrics by Tobin (1958) and have since been a topic of active research (DeMaris, 2004; Greene, 2012; Wooldridge, 2002).

Censored regression models come in a variety of different forms: parametric, semi-parametric, and non-parametric. The earliest models were mostly parametric whereas nowadays interest is in the development of semi- and non-parametric models. The parametric modeling strategy is often not applicable since econometric data sets often have non-linear relationships and the underlying data generating mechanism is not precisely known. Gaussian processes (GPs) (Rasmussen and Williams, 2006) are Bayesian non-parametric graphical models that provide state-of-the-art performance in regression tasks.

Gaussian processes are an extension of multivariate Gaussian random variables to infinite index sets that define a probability distribution over functions and inference takes place directly in the space of functions. An interesting property of Gaussian processes is that they provide a full predictive distribution, i.e., a mean estimate and an uncertainty estimate for the mean.

The contribution of this paper is an extension of Gaussian process regression models to data in which some observations are subject to censoring. Since the model is not analytically tractable we use an approximation for computing the posterior. In (Ertin, 2007) a Laplace approximation was considered. Since the censored Gaussian process model is, however, closely related to the probit model for which Expectation propagation was found to be the method of choice (Nickisch and Rasmussen, 2008) we use Expectation propagation to perform approximate inference on the model. As the model satisfies integrability with respect to Gaussian measures all necessary derivations for EP can be done analytically leading to a numerically stable algorithm. The method is validated on a benchmark data set. The rest of the paper is structured as follows. Section 2 describes the Bayesian framework. Section 3 describes experimental results. Section 4 describes related work. Section 5 gives conclusions.

## 2 Bayesian framework

In the following we denote vectors  $\mathbf{x}$  and matrices  $\mathbf{K}$  with bold-face type and their components with regular type, i.e.,  $x_i$ ,  $K_{ij}$ . With  $\mathbf{x}^T$  we denote the transpose of the vector  $\mathbf{x}$ . Let  $\mathbf{x} \in \mathbb{R}^D$  be a vector of explanatory variables,  $y^* \in \mathbb{R}$  a response variable. We consider the problem of learning a function  $f : \mathbb{R}^D \rightarrow \mathbb{R}$  from a set  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$  of  $n$  observations, where  $y$  is a censored version of  $y^*$ :

$$y = \begin{cases} l & \text{if } y^* \leq l \\ y^* & \text{if } l < y^* < u \\ u & \text{if } y^* \geq u \end{cases} \quad (1)$$

where  $l, u \in \mathbb{R}$  with  $l < u$ , are lower and upper thresholds, respectively. Hence, values in a certain range are transformed to (or reported as) a single value. Censoring represents a limitation in the response variable  $y^*$  of interest, but the explanatory variable  $x$  is assumed to be fully observable.<sup>1</sup> When  $y = l$  we only know a lower bound on the target value  $y^* \in [l, \infty)$  which is referred to as being *right censored*. When  $y = u$  we only know an upper bound on the target value  $y^* \in (-\infty, u]$  which is referred to as being *left censored*.

### 2.1 Prior

We assume that the latent values  $y^* = f(\mathbf{x})$  are the realization of a zero-mean Gaussian process (Rasmussen and Williams, 2006), which can then be fully specified by the covariance matrix. The covariance matrix can be specified in terms of a kernel function  $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ . An often used kernel function is the Gaussian kernel

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{d=1}^D \frac{(x_d - x'_d)^2}{\ell_d^2}\right) \quad (2)$$

where  $x_d$  denotes the  $d$ -th element of  $\mathbf{x}$ , and  $\boldsymbol{\theta} = \{\sigma_f^2, \ell_1, \dots, \ell_D\}$  hyperparameters with  $\sigma_f^2$  specifying the signal variance and  $\ell_d$  specifying how much the function can vary in the  $d$ -th element of the explanatory variables. For brevity

<sup>1</sup>In contrast, if the explanatory variable is not observable, i.e., missing, the data is said to be truncated.

we often omit the dependence on  $\boldsymbol{\theta}$  in formulas. The kernel function leads to a multivariate Gaussian prior of latent function values  $\{f(\mathbf{x}_i)\}$

$$p(\mathbf{f}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\mathbf{K}|^{\frac{D}{2}}} \exp\left(-\frac{1}{2} \mathbf{f}^T \mathbf{K}^{-1} \mathbf{f}\right) \quad (3)$$

where  $\mathbf{K}$  is the  $n \times n$  covariance matrix with  $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ .

### 2.2 Likelihood

To capture the relations in (1) we can define an ideal likelihood for noise-free cases:

$$p_{id}(y|f(x)) = \begin{cases} 1 & \text{if } y = l, f(x) \leq l, \\ & \text{or } l < y = f(x) < u, \\ & \text{or } y = u, f(x) \geq u \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

To tolerate noise in the explanatory variables or response variables we assume that the unobserved latent function values are contaminated with noise and that these noisy values are then censored. A common choice is Gaussian noise with zero mean and unknown variance  $\sigma^2$ .<sup>2</sup> We use  $\mathcal{N}(\delta|\mu, \sigma^2)$  to denote a Gaussian random variable  $\delta$  with mean  $\mu$  and variance  $\sigma^2$  and use  $\phi(\cdot)$ ,  $\Phi(\cdot)$  to denote the standard normal density and cdf, respectively. Under these assumptions the following equations hold

$$\begin{aligned} p(y = l|f) &= \int p_{id}(y = l|f + \delta) \mathcal{N}(\delta|0, \sigma^2) d\delta \\ &= \Phi\left(\frac{l - f}{\sigma}\right) = 1 - \Phi\left(\frac{f - l}{\sigma}\right) \\ p(y = y^*|f) &= \mathcal{N}(y|f, \sigma^2) = \frac{1}{\sigma} \phi\left(\frac{y - f}{\sigma}\right) \\ p(y = u|f) &= \Phi\left(\frac{f - u}{\sigma}\right) \end{aligned} \quad (5)$$

<sup>2</sup>In principle, one could assume any distribution for the noise on the latent functions. Another often used assumption is heteroscedastic Gaussian noise.

The likelihood therefore becomes a mixture of Gaussian and probit likelihood terms:

$$\begin{aligned} L = \prod_{i=1}^n p(y_i|f_i) &= \prod_{y_i=l} \left[ 1 - \Phi\left(\frac{f_i - l}{\sigma}\right) \right] \\ &\quad \prod_{l < y_i < u} \left[ \frac{1}{\sigma} \phi\left(\frac{y_i - f_i}{\sigma}\right) \right] \quad (6) \\ &\quad \prod_{y_i=u} \left[ \Phi\left(\frac{f_i - u}{\sigma}\right) \right] \end{aligned}$$

This likelihood is well known in the literature as a Tobit likelihood (DeMaris, 2004; Greene, 2012), or a type I Tobit model according to the taxonomy of (Amemiya, 1984). The model can nowadays be estimated by many software packages, however, these are typically limited by a *parametric, linear form*.

### 2.3 Posterior

The posterior distribution over the latent variables is given by Bayes' theorem as a product of a normalization term, the prior, and the likelihood

$$p(\mathbf{f}|X, y) = \frac{1}{Z} p(\mathbf{f}|X) \prod_{i=1}^n p(y_i|f_i) \quad (7)$$

where the normalization term is called the evidence or marginal likelihood

$$Z = p(y|X) = \int p(\mathbf{f}|X) \prod_{i=1}^n p(y_i|f_i) d\mathbf{f} \quad (8)$$

The Tobit likelihood in (6) makes the posterior in (7) analytically intractable and one has to resort to approximation techniques for computing the posterior. Below we describe the Expectation propagation method.

#### 2.3.1 Expectation Propagation

Expectation propagation (EP) (Minka, 2001) is a deterministic approximate inference method that tackles the non-analytic nature of the posterior in (7) by approximating the likelihood by a local likelihood approximation using an unnormalized Gaussian function in the latent variable  $f_i$

$$\begin{aligned} p(y_i|f_i) &\simeq t_i(f_i|\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) \\ &= \tilde{Z}_i \mathcal{N}(f_i|\tilde{\mu}_i, \tilde{\sigma}_i^2) \quad (9) \end{aligned}$$

where  $t_i$  is called the  $i$ -th *site* with site parameters  $\tilde{Z}_i$ ,  $\tilde{\mu}_i$ , and  $\tilde{\sigma}_i^2$ , which we often abbreviate as  $t_i(f_i)$  or simply  $t_i$ .

The product of the  $n$  likelihood approximations  $t_i$  can be expressed as

$$\prod_{i=1}^n t_i(f_i|\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) = \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \prod_{i=1}^n \tilde{Z}_i \quad (10)$$

where  $\tilde{\boldsymbol{\mu}}$  is the vector of  $\tilde{\mu}_i$  and  $\tilde{\boldsymbol{\Sigma}}$  is diagonal with  $\tilde{\Sigma}_{ii} = \tilde{\sigma}_i^2$ . The posterior  $p(\mathbf{f}|\mathcal{D})$  is approximated by  $q(\mathbf{f}|\mathcal{D})$  where

$$\begin{aligned} q(\mathbf{f}|\mathcal{D}) &= \frac{1}{Z_{EP}} p(\mathbf{f}) \prod_{i=1}^n t_i(f_i|\tilde{Z}_i, \tilde{\mu}_i, \tilde{\sigma}_i^2) \\ &= \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (11) \\ \boldsymbol{\mu} &= \boldsymbol{\Sigma} \tilde{\boldsymbol{\mu}} \\ \boldsymbol{\Sigma} &= (\mathbf{K}^{-1} + \tilde{\boldsymbol{\Sigma}}^{-1})^{-1} \end{aligned}$$

using the product rule of Gaussian distributions and where  $Z_{EP}$  is the approximation of the EP algorithm to the marginal likelihood in (8).

The EP algorithm iteratively updates the  $t_i$  approximations sequentially. This is done by removing the  $i$ -th term from the approximate posterior, resulting in a *cavity distribution*, which is then combined with the  $i$ -th exact likelihood term. Finally a Gaussian approximation to the non-Gaussian marginal is computed, which is then used to update  $t_i$ .

EP defines the cavity distribution  $q_{\setminus i}$ , which is a Gaussian distribution, as

$$\begin{aligned} q_{\setminus i} &\propto \int p(\mathbf{f}) \prod_{j \neq i} t_j(f_j) = \mathcal{N}(\mu_{\setminus i}, \sigma_{\setminus i}^2) \\ \mu_{\setminus i} &= \sigma_{\setminus i}^2 (\sigma_i^{-2} \mu_i - \tilde{\sigma}_i^{-2} \tilde{\mu}_i) \\ \sigma_{\setminus i}^2 &= (\sigma_i^{-2} - \tilde{\sigma}_i^{-2})^{-1} \quad (12) \end{aligned}$$

We then continue to find the new unnormarized Gaussian marginal which best approximates the product of the cavity distribution and the exact likelihood

$$\hat{q}(f_i) = \tilde{Z}_i \mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i^2) \simeq q_{\setminus i} p(y_i|f_i) \quad (13)$$

When minimizing  $KL(p(x)||q(x))$  in the case of a Gaussian distribution  $q(x)$ , the minimum is

obtained by matching the zeroth, first, and second moment. The parameters of the local likelihood approximation  $t_i$  can be computed using

$$\begin{aligned}\tilde{\mu}_i &= \tilde{\sigma}_i^2 (\hat{\sigma}_i^{-2} \hat{\mu}_i - \sigma_{\setminus i}^{-2} \mu_{\setminus i}) \\ \tilde{\sigma}_i^2 &= (\hat{\sigma}_i^{-2} - \sigma_{\setminus i}^{-2})^{-1} \\ \tilde{Z}_i &= \hat{Z}_i \sqrt{2\pi(\sigma_{\setminus i}^2 + \tilde{\sigma}_i^2)} \exp\left(\frac{1}{2} \frac{(\mu_{\setminus i} - \tilde{\mu}_i)^2}{(\sigma_{\setminus i}^2 + \tilde{\sigma}_i^2)}\right)\end{aligned}$$

where  $\hat{Z}_i$ ,  $\hat{\mu}_i$ , and  $\hat{\sigma}_i^2$  are the moments that minimize the KL divergence which are described in the next section for the Tobit likelihood.

### 2.3.2 Tobit Moments

The minimum of the KL divergence is obtained when the moments of the two Gaussian distributions match (Minka, 2001). We thus need to compute the zeroth, first, and second order moments

$$\begin{aligned}\hat{Z}_i &= \int q_{\setminus i} t_i \mathrm{d}f_i \\ \hat{\mu}_i &= \mathbf{E}_q[f_i] \\ \hat{\sigma}_i^2 &= \mathbf{E}_q[(f_i - \mathbf{E}_q[f_i])^2]\end{aligned}\tag{14}$$

where  $q = \hat{Z}_i^{-1} q_{\setminus i} t_i$ , which in the case of the Tobit likelihood in (6) can be computed analytically (cf. (Rasmussen and Williams, 2006)). The moments depend on whether the observation is left censored, right censored, or non-censored. Let  $z_i^m = \frac{\mu_i - m}{\sqrt{\sigma^2 + \sigma_{\setminus i}^2}}$  for  $m \in \{l, u\}$ .

If  $y_i = l$  then

$$\begin{aligned}\hat{Z}_i &= 1 - \Phi(z_i^l) \\ \hat{\mu}_i &= \mu_{\setminus i} + \frac{\sigma_{\setminus i}^2 \mathcal{N}(z_i^l)}{(1 - \Phi(z_i^l)) \sqrt{\sigma^2 + \sigma_{\setminus i}^2}} \\ \hat{\sigma}_i^2 &= \hat{Z}_i^{-1} [(\sigma_{\setminus i}^2 + \mu_{\setminus i}^2) - \hat{Z}_{i(p)} (\hat{\sigma}_{i(p)}^2 + \hat{\mu}_{i(p)}^2)] - \hat{\mu}_i^2\end{aligned}$$

where  $\hat{Z}_{i(p)}$ ,  $\hat{\sigma}_{i(p)}^2$ ,  $\hat{\mu}_{i(p)}^2$  are the zeroth, first, and second order moments of a probit likelihood

$\Phi(\frac{f-l}{\sigma})$ . If  $l < y_i < u$  then

$$\begin{aligned}\hat{Z}_i &= \frac{1}{\sqrt{2\pi(\sigma^2 + \sigma_{\setminus i}^2)}} \exp\left(-\frac{1}{2} \frac{(y_i - \mu_{\setminus i})^2}{(\sigma^2 + \sigma_{\setminus i}^2)}\right) \\ \hat{\mu}_i &= \mu_{\setminus i} + \sigma_{\setminus i}^2 \left( \frac{y_i - \mu_{\setminus i}}{\sigma^2 + \sigma_{\setminus i}^2} \right) \\ \hat{\sigma}_i^2 &= \sigma_{\setminus i}^2 - \sigma_{\setminus i}^4 \frac{1}{(\sigma^2 + \sigma_{\setminus i}^2)}\end{aligned}$$

If  $y_i = u$  then

$$\begin{aligned}\hat{Z}_i &= \Phi(z_i^u) \\ \hat{\mu}_i &= \mu_{\setminus i} + \frac{\sigma_{\setminus i}^2 \mathcal{N}(z_i^u)}{\Phi(z_i^u) \sqrt{\sigma^2 + \sigma_{\setminus i}^2}} \\ \hat{\sigma}_i^2 &= \sigma_{\setminus i}^2 - \frac{\sigma_{\setminus i}^4 \mathcal{N}(z_i^u)}{(\sigma^2 + \sigma_{\setminus i}^2) \Phi(z_i^u)} \left( z_i^u + \frac{\mathcal{N}(z_i^u)}{\Phi(z_i^u)} \right)\end{aligned}$$

### 2.4 Prediction

The approximate predictive distribution  $q(f_*)$  given an input  $\mathbf{x}_*$  results from the approximation to the posterior distribution given by EP (11) and can be written as

$$\begin{aligned}q(f_*) &= \int p(f_* | \mathbf{x}_*, \mathcal{D}, \mathbf{f}) q(\mathbf{f}) \mathrm{d}\mathbf{f} \\ &= \mathcal{N}(\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{k}_*^T (\mathbf{K} + \tilde{\Sigma})^{-1} \tilde{\boldsymbol{\mu}} \\ \sigma_*^2 &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (\mathbf{K} + \tilde{\Sigma})^{-1} \mathbf{k}_*\end{aligned}\tag{15}$$

where  $\mathbf{k}_* = [k(\mathbf{x}_1, \mathbf{x}_*), \dots, k(\mathbf{x}_n, \mathbf{x}_*)]^T$ .

### 2.5 Marginal Likelihood

The approximation of the EP algorithm to the marginal likelihood in (8) is given by

$$\begin{aligned}p(\mathcal{D} | \boldsymbol{\theta}) &= \int p(\mathbf{f} | \boldsymbol{\theta}_1) \prod_{i=1}^n p(y_i | f_i, \boldsymbol{\theta}_2) \mathrm{d}\mathbf{f} \\ &\approx \int p(\mathbf{f} | \boldsymbol{\theta}_1) \prod_{i=1}^n t_i(f_i) \mathrm{d}\mathbf{f}\end{aligned}\tag{16}$$

where  $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$  are the hyperparameters with  $\boldsymbol{\theta}_1$  the covariance hyperparameters and  $\boldsymbol{\theta}_2$  the likelihood hyperparameters. The hyperparameters can be optimized by minimizing the

negative log marginal likelihood using gradient descent where

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}_1} \log p(\mathcal{D}|\boldsymbol{\theta}) &= -\frac{1}{2} \frac{\partial}{\partial \boldsymbol{\theta}_1} (\log |\boldsymbol{\Sigma} + \mathbf{K}| + \boldsymbol{\mu}^T (\boldsymbol{\Sigma} + \mathbf{K})^{-1} \boldsymbol{\mu}) \\ \frac{\partial}{\partial \boldsymbol{\theta}_2} \log p(\mathcal{D}|\boldsymbol{\theta}) &= \frac{\partial}{\partial \boldsymbol{\theta}_2} \sum_{i=1}^n \log \hat{Z}_i \end{aligned}$$

### 3 Experiments

In this section we compare the standard Gaussian process model with the Tobit Gaussian process model, which was implemented in the publicly available GPstuff toolbox (Vanhatalo et al., 2011) on both artificial data and real data. In order to compare the predictive performance of the models we cast the analysis as a ranking problem, which is an elegant way of dealing with the censoring of the data. Two predictions can be ordered if (1) both values are non-censored, (2) if the non-censored value of one is smaller than the left censored value of the other, (3) if the non-censored value of one is larger than the right censored value of the other, or (4) if the right censored value of one is larger than the left censored value of the other. These four possible ordering relations are illustrated in Figure 1.

For these reasons the *concordance index* or *c-index* is often used as a performance measure for comparing models with censored data (Harrell Jr., 2001). The c-index can be interpreted as the fraction of all pairs of inputs whose predicted values are correctly ordered among all inputs that can be ordered. The c-index can be written as

$$c(\mathcal{D}, \mathcal{G}, f) = \frac{1}{|\mathcal{E}|} \sum_{\mathcal{E}_{ij}} \mathbf{1}_{f(x_i) < f(x_j)} \quad (17)$$

where  $\mathcal{G} = (X, \mathcal{E})$  is the order graph with edges according to the four criteria given above,  $|\mathcal{E}|$  is the number of edges in  $\mathcal{E}$ , and  $\mathbf{1}_{x < y} = 1$  if  $x < y$ , 0 otherwise, is an indicator function.

The concordance index is a generalization of the Wilcoxon-Mann-Whitney statistics (Wilcoxon, 1945; Mann and Whitney, 1947) for which  $c = 1$  indicates a perfect predictive model

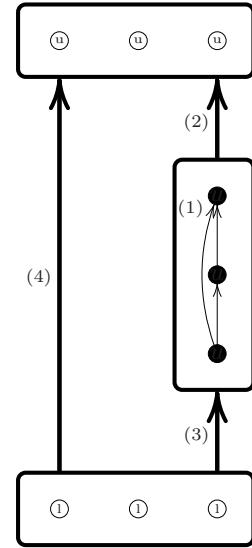


Figure 1: Four possible ordering relations when data is censored. The white/open circles represent censored data points and are labeled  $u$  when it is left censored or 1 when right censored. Black/closed dots represent non-censored data points. Arrows between (groups of) points represents which data points can possibly be ordered where the number refers to the enumeration used in the article.

and  $c = 0.5$  indicates a random predictive model and is thus an AUC-like metric for the scenario of censored data.

#### 3.1 Artificial data

To illustrate the behaviour of the standard GP model and the Tobit GP model, we first illustrate their behaviour on a simple 1-dimensional function. We created a data set of 30 equally spaced inputs in  $[0, 1]$  and outputs using

$$f(x) = (6x - 2)^2 \sin(2(6x - 2)) \quad (18)$$

which were contaminated with Gaussian noise with zero mean and variance 0.1. To censor 40% of the observations we calculated the 40th percentile of the sample distribution and used the corresponding value of  $l = -0.2265$  as lower threshold. We trained a standard GP model and a Tobit GP model on the full data set of 30 samples including censored observations, as

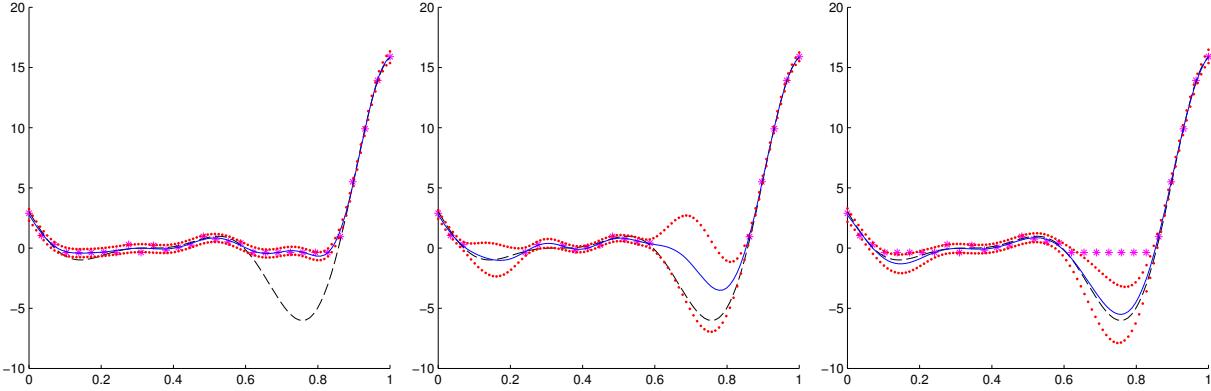


Figure 2: Regression with censored data. Left panel: standard GP model on data with censored samples. Middle panel: standard GP model on data where censored samples were removed. Right panel: Tobit GP model on data with censored samples. The  $*$  denotes observations (possibly noisy and/or censored). Dashed line denotes the underlying true function. Solid line denotes the mean model prediction. Dotted line denotes two standard deviations from the mean.

well as a standard GP model on the data set of 18 samples of non-censored observations.

The results are shown in Figure 2. The left panel shows a standard GP model on the data set of 30 observations which clearly shows a bad predictive model of the underlying true function as it tries to match the mean prediction with the censored observations and also severely undervalues the model uncertainty in the censored areas. The middle panel shows a standard GP model on the data of 18 non-censored samples, which performs a bit better than the standard GP model on the full data set both in terms of the mean prediction and the uncertainty of the mean prediction. The model, however, throws out information and is not able to predict that the underlying true function lies strictly below the censoring threshold. The right panel shows the Tobit GP model which shows the best mean prediction, has a lower predictive uncertainty, and in the censored areas gives predictions that lie strictly below the censoring threshold.

We report the results of 10-fold crossvalidation of both the standard GP and Tobit GP in Table 1. As the number of censored observations is quite large, i.e., 40% of the data, we observe a large increase in the c-index for the Tobit GP model. Since we know the true underlying latent function we also report the root

mean squared error

$$RMSE(t, f) = \sqrt{\frac{1}{n} \sum_{i=1}^n (t_i - f_i)^2} \quad (19)$$

and the mean absolute error

$$MAE(t, f) = \frac{1}{n} \sum_{i=1}^n |t_i - f_i| \quad (20)$$

in Table 1 for both the standard GP and Tobit GP models.

Table 1: Concordance results and error measures artificial data.

|         | GP   | Tobit GP |
|---------|------|----------|
| c-index | 0.73 | 0.95     |
| RMSE    | 2.21 | 0.72     |
| MAE     | 1.42 | 0.60     |

### 3.2 Housing data

We validated the Gaussian process Tobit model on the ‘housing’ benchmark data set obtained from the UCI machine learning repository. Originally the housing data set was used to investigate methodological issues related to the use of housing data to estimate the demand for

clean air (Harrison and Rubinfeld, 1978). The data consists of 506 observations on 14 real-valued variables. The objective is to predict the median value of owner-occupied homes from the remaining variables. In (Gilley and Pace, 1996), the data has been checked against the original census data and it was discovered that the Census Bureau censored tracts whose median value was over \$50.000. Hence, all tracts with a median value greater than \$50.000 appear as \$50.000. In total 16 observations were censored.

We report the results of 10-fold crossvalidation of both the standard GP and Tobit GP in Table 2 averaged over 10 runs. Although the number of censored observations is quite small, i.e., only 3.2% of the data, we observe an increase in the c-index for the Tobit GP model. We also see an improvement in predictive performance for the Tobit GP model when using the Expectation algorithm (EP) instead of the Laplace approximation (LA) of (Ertin, 2007).

Table 2: Concordance results housing data (mean c-index and standard deviation).

| method        | c-index           |
|---------------|-------------------|
| GP            | $0.866 \pm 0.003$ |
| Tobit-GP (LA) | $0.879 \pm 0.008$ |
| Tobit-GP (EP) | $0.892 \pm 0.007$ |

## 4 Related work

In recent years researchers have striven to devise regression techniques that do not rely on the classical assumptions of parametric methods. In this paper we extended the Gaussian process framework, a principled non-parametric Bayesian framework for regressions tasks, to data subject to censoring. Since the model is no longer analytically tractable approximations are needed to compute the posterior. In this paper we considered the Expectation propagation algorithm, which was shown to outperform the Laplace approximation considered in (Ertin, 2007). In (Hutter et al., 2011) a different approach is followed to handle censored data. Hutter et al., uses a random forest, initialized using

non-censored data only, which is iteratively improved using an EM algorithm. Their method samples data from the predictive distribution to fill in samples for censored observations. However, since the predictive distribution is not guaranteed to lie strictly below (or above) the censoring threshold (cf. Figure 2), additional countermeasures, like truncating the predictive distribution, are needed to ensure the data generated satisfies the censoring constraints.

In this paper we considered common lower and upper censoring thresholds for each data point, however, the model can easily be extended to allow for data points with individual lower and/or upper thresholds. This would allow the method to be used for the analysis of survival times (Shivaswamy et al., 2007) in which censoring is extremely common. The concordance index can still be used as a performance measure as given in Equation (17), but the situation depicted in Figure 1 changes. Since data points have individual censoring thresholds the ordering relations in Figure 1 do not longer hold and some may become incomparable (Shivaswamy et al., 2007).

In the likelihood model we assumed that noise was Gaussian distributed. If this assumption is violated this can have considerable impact on the predictive performance of the model (Greene, 2012). A more general approach would be to allow for heteroscedastic Gaussian distributed noise by using a second Gaussian process to model the noise process (Kersting et al., 2007). One could follow a similar EP approach as in (Muñoz-González et al., 2011), although with respect to this second Gaussian process the model would then no longer satisfy integrability with respect to Gaussian measures.

## 5 Conclusion

In this paper we have presented an extension to Gaussian process regression to handle data subject to censoring. To tackle the non-analytic nature of the posterior we used Expectation propagation to perform approximate inference on the model which was shown to outperform the Laplace approximation. Directions for fur-

ther work include among others handling heteroscedastic noise, multi-variate Tobit models, and extending the model to other domains such as survival analysis.

The software was implemented in the publicly available GPstuff toolbox (Vanhatalo et al., 2011) and is freely available from the first authors website as well as some supplementary material describing implementation details.

## References

- T. Amemiya. 1984. Tobit models: A survey. *Journal of Econometrics*, 24(1-2):3–61.
- A. DeMaris. 2004. *Regression With Social Data: Modeling Continuous and Limited Response Variables*. Wiley Series in Probability and Statistics. Wiley-Interscience.
- E. Ertin. 2007. Gaussian process models for censored sensor readings. In *Proceedings of the 2007 IEEE/SP 14th Workshop on Statistical Signal Processing*, SSP ’07, pages 665–669, Washington, DC, USA. IEEE Computer Society.
- O. W. Gilley and R. K. Pace. 1996. On the Harrison and Rubinfeld data. *Journal of Environmental Economics and Management*, 31:403–405.
- W. H. Greene. 2012. *Econometric Analysis*. Prentice Hall, 7th edition.
- F. E. Harrell Jr. 2001. *Regression Modeling Strategies, With applications to linear models, logistic regression, and survival analysis*. Springer.
- D. Harrison and D. L. Rubinfeld. 1978. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5:81–102.
- F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2011. Bayesian optimization with censored response data. In *NIPS workshop on Bayesian optimization, sequential experimental design, and bandits*.
- K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. 2007. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th international conference on Machine learning*, ICML ’07, pages 393–400, New York, NY, USA. ACM.
- H. B. Mann and D. R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18(1):50–60.
- T. Minka. 2001. *A family of approximation methods for approximate Bayesian inference*. Ph.D. thesis, MIT.
- L. Muñoz-González, M. Lázaro-Gredilla, and A. R. Figueiras-Vidal. 2011. Heteroscedastic Gaussian process regression using expectation propagation. In *Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.
- H. Nickisch and C. E. Rasmussen. 2008. Approximations for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 9:2035–2078, October.
- C. E. Rasmussen and C. K. I. Williams. 2006. *Gaussian processes for machine learning*. MIT Press, Cambridge, MA.
- P. K. Shivaswamy, W. Chu, and M. Jansche. 2007. A support vector approach to censored targets. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM ’07, pages 655–660, Washington, DC, USA. IEEE Computer Society.
- J. Tobin. 1958. Estimation of relationships for limited dependent variables. *Econometrica*, 26:24–36.
- J. Vanhatalo, J. Riihimäki, J. Hartikainen, and A. Vehtari. 2011. Bayesian modeling with Gaussian processes using the MATLAB toolbox GPstuff. submitted.
- F. Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.
- J. M. Wooldridge. 2002. *Econometric Analysis of Cross Section and Panel Data*. Mit Press.

# Two Optimal Strategies for Active Learning of Causal Models from Interventions

Alain Hauser and Peter Bühlmann

ETH Zürich, Switzerland

{hauser, buhlmann}@stat.math.ethz.ch

## Abstract

From observational data alone, a causal DAG is in general only identifiable up to Markov equivalence. Interventional data generally improves identifiability; however, the gain of an intervention strongly depends on the intervention target, i.e., the intervened variables. We present active learning strategies calculating optimal interventions for two different learning goals. The first one is a greedy approach using single-vertex interventions that maximizes the number of edges that can be oriented after each intervention. The second one yields in polynomial time a minimum set of targets of arbitrary size that guarantees full identifiability. This second approach proves a conjecture of Eberhardt (2008) indicating the number of unbounded intervention targets which is sufficient and in the worst case necessary for full identifiability. We compare our two active learning approaches to random interventions in a simulation study.

## 1 Introduction

Causal relationships between random variables are usually modeled by directed acyclic graphs (DAGs), where an arrow between two random variables,  $X \rightarrow Y$ , reveals the former ( $X$ ) as a *direct* cause of the latter ( $Y$ ). From observational data alone (i.e. *passively* observed data from the undisturbed system), directed graphical models are only identifiable up to Markov equivalence, and arrow directions (which are crucial for the causal interpretation) are in general not identifiable. Without the assumption of specific functional model classes and error distributions (Peters et al., 2011), the only way to improve identifiability is to use interventional data for estimation, i.e. data produced under a perturbation of the system in which one or several random variables are forced to specific values, irrespective of the original causal parents.

The investigation of observational Markov equivalence classes has a long tradition in the literature (Verma and Pearl, 1990; Andersson et al., 1997; Spirtes et al., 2000). In a recent paper, Hauser and Bühlmann (2012) presented a graph-theoretic characterization of interven-

tional Markov equivalence classes for a given set of interventions (possibly affecting several variables simultaneously). In this paper, we present two active learning strategies for finding valuable interventions: one that greedily optimizes the number of orientable edges with single-vertex interventions, and one that minimizes the number of interventions at arbitrarily many vertices to attain full identifiability.

Several approaches for actively learning causal models have been proposed during the last decade. Our method for finding intervention targets of *unbounded size* is closely related to the approach of Eberhardt (2008). In contrast to his procedure, our algorithm has a polynomial time complexity; furthermore, we prove his conjecture on the number of intervention experiments sufficient and in the worst case necessary for fully identifying a causal model. He and Geng (2008) presented a method to find a (nearly) minimal set of single-vertex interventions which guarantee the orientability of all undirected edges of an observational Markov equivalence class. In contrast to their approach, we proceed in a greedy way which

results in a smaller or at most equal number of single-vertex interventions to be performed. Tong and Koller (2001) finally proposed a Bayesian active learning strategy that minimizes an expected loss, in contrast to our frequentist methods.

This paper is organized as follows: in Sect. 2, we specify our notation of causal models and formalize our learning goals. In Sect. 3, we summarize graph-theoretic background material upon which our active learning algorithms, presented in Sect. 4, are based. In Sect. 5, we evaluate our algorithms in a simulation study.

## 2 Model

### 2.1 Causal Calculus

We consider a causal model on  $p$  random variables  $(X_1, \dots, X_p)$  described by a DAG  $D$ . Formally, a causal model is a pair  $(D, f)$ , where  $D$  is a DAG on the vertex set  $[p] := \{1, \dots, p\}$  which encodes the **Markov property** of the (observational) density  $f$ :  $f(x) = \prod_{i=1}^p f(x_i | x_{\text{pa}_D(i)})$ ;  $\text{pa}_D(i)$  denotes the parent set of vertex  $i$  (see also Sect. 3). Unless stated otherwise, all graphs in this paper are assumed to have the vertex set  $[p]$ .

Beside the conditional independence relations of the observational density implied by the Markov property, a causal model also makes statements about effects of **interventions**. We consider **stochastic interventions** (Korb et al., 2004) modeling the effect of setting or forcing one or several random variables  $X_I := (X_i)_{i \in I}$ , where  $I \subset [p]$  is called the **intervention target**, to the value of *independent* random variables  $U_I$ . Extending the `do()` operator (Pearl, 1995) to stochastic interventions, we denote the **interventional density** of  $X$  under such an intervention by

$$f(x | \text{do}_D(X_I = U_I)) := \prod_{i \notin I} f(x_i | x_{\text{pa}_D(i)}) \prod_{i \in I} \tilde{f}(x_i),$$

where  $\tilde{f}$  is the density of  $U_I$  on  $\mathcal{X}_I$ . By denoting with  $I = \emptyset$  and using the convention  $f(x | \text{do}(X_\emptyset = U_\emptyset)) = f(x)$ , we also encompass the observational case as an intervention target. The interventional density  $f(x | \text{do}_D(X_I = U_I))$

has the Markov property of the **intervention graph**  $D^{(I)}$ , the DAG that we get from  $D$  by removing all arrows pointing to vertices in  $I$ .

We consider experiments based on datasets originating from *multiple* interventions. The **family of targets**  $\mathcal{I} \subset \mathcal{P}([p])$ , where  $\mathcal{P}([p])$  denotes the power set of  $[p]$ , lists all (distinct) intervention targets used in an experiment. A family of targets  $\mathcal{I} = \{\emptyset, \{3\}, \{1, 4\}\}$  e.g. characterizes an experiment in which observational data as well as data originating from an intervention at  $X_3$  and data originating from a (simultaneous) intervention at  $X_1$  and  $X_4$  are measured. In the whole paper,  $\mathcal{I}$  always stands for a family of targets with the property that for each vertex  $a \in [p]$ , there is some target  $I \in \mathcal{I}$  in which  $a$  is *not* intervened ( $a \notin I$ ). This is e.g. the case when observational data is available ( $\emptyset \in \mathcal{I}$ ). Two DAGs  $D_1$  and  $D_2$  are called  **$\mathcal{I}$ -Markov equivalent** ( $D_1 \sim_{\mathcal{I}} D_2$ ) if they are statistically indistinguishable under an experiment consisting of interventions at the targets in  $\mathcal{I}$ ; we refer to Hauser and Bühlmann (2012) for a more formal treatment.

**Theorem 1** (Hauser and Bühlmann (2012)). *Two DAGs  $D_1$  and  $D_2$  are  $\mathcal{I}$ -Markov equivalent if and only if*

- (i)  *$D_1$  and  $D_2$  have the same skeleton and the same v-structures (that is, induced subgraphs of the form  $a \rightarrow b \leftarrow c$ ), and*
- (ii)  *$D_1^{(I)}$  and  $D_2^{(I)}$  have the same skeleton for all  $I \in \mathcal{I}$ .*

An  $\mathcal{I}$ -Markov equivalence class of a DAG  $D$  is uniquely represented by its  **$\mathcal{I}$ -essential graph**  $\mathcal{E}_{\mathcal{I}}(D)$  (Hauser and Bühlmann, 2012). This partially directed graph has the same skeleton as  $D$ ; a directed edge in  $\mathcal{E}_{\mathcal{I}}(D)$  represents  **$\mathcal{I}$ -essential** arrows, i.e. arrows that have the same orientation in all DAGs of the equivalence class; an undirected edge represents arrows that have different orientations in different DAGs of the equivalence class. The concept of  $\mathcal{I}$ -essential graphs generalizes the one of CPDAGs which is well-known in the observational case (Spirtes et al., 2000; Andersson et al., 1997). We denote the  $\mathcal{I}$ -Markov equivalence class corresponding to an  $\mathcal{I}$ -essential graph  $G$  by  $\mathbf{D}(G)$ .

## 2.2 Active Learning

Assume  $G$  is an  $\mathcal{I}$ -essential graph estimated from interventional data produced under the different interventions in  $\mathcal{I}$ . We consider two different greedy active learning approaches. In one step, the first one computes a single-vertex intervention that maximizes the number of orientable edges, while the second one computes an intervention target of arbitrary size that maximally reduces the clique number, i.e. the size of the largest clique of undirected edges (see Sect. 3). The motivation for the first approach is the attempt to quickly improve the identifiability of causal models with interventions at few variables; the motivation for the second approach is the conjecture of Eberhardt (2008) (which we prove in Cor. 2) stating that maximally reducing the clique number after each intervention yields full identifiability of causal models with a minimal number of interventions.

Formally, our two algorithms yield a solution to the following problems. The first one, called OPTSINGLE, computes a vertex

$$v = \arg \min_{v' \in [p]} \max_{D \in \mathbf{D}(G)} \xi(\mathcal{E}_{\mathcal{I} \cup \{v'\}}(D)) , \quad (1)$$

where  $\xi(H)$  denotes the number of undirected edges in a graph  $H$ . The second algorithm, called OPTUNB, computes a set

$$I = \arg \min_{I' \subset [p]} \max_{D \in \mathbf{D}(G)} \omega(\mathcal{E}_{\mathcal{I} \cup \{I'\}}(D)) , \quad (2)$$

where  $\omega(H)$  denotes the clique number of  $H$  (see also Sect. 3). The key ingredients for the efficiency of OPTSINGLE (Alg. 2) and OPTUNB (Alg. 3) are implementations that minimize the objective functions of Eq. (1) and (2), resp., without enumerating all DAGs in the equivalence class represented by  $G$ . Graph theoretic results upon which our implementations are based are summarized in the next section.

## 3 Graph Theoretic Background

A **graph** is a pair  $G = (V, E)$ , where  $V$  is a set of vertices and  $E \subset (V \times V) \setminus \{(a, a) | a \in V\}$  is a set of edges. We always assume  $V = [p] := \{1, 2, \dots, p\}$  and let the vertices of a graph represent the  $p$  random variables  $X_1, \dots, X_p$ .

An edge  $(a, b) \in E$  with  $(b, a) \in E$  is called **undirected** and denoted by  $a — b$ , whereas an edge  $(a, b) \in E$  with  $(b, a) \notin E$  is called **directed** and denoted by  $a \rightarrow b$ .  $G$  is called directed if all its edges are directed (or undirected, resp.). A **cycle** of length  $k \geq 2$  is a sequence of  $k$  distinct vertices of the form  $(a_0, a_1, \dots, a_k = a_0)$  such that  $(a_{i-1}, a_i) \in E$  for  $i \in \{1, \dots, k\}$ ; the cycle is **directed** if at least one edge is directed.

For a subset  $A \subset V$  of the vertices of  $G$ , the **induced subgraph** on  $A$  is  $G[A] := (A, E[A])$ , where  $E[A] := E \cap (A \times A)$ . A **v-structure** is an induced subgraph of  $G$  of the form  $a \rightarrow b \leftarrow c$ . The **skeleton** of a graph  $G$  is the undirected graph  $G^u := (V, E^u)$ ,  $E^u := E \cup \{(a, b) | (b, a) \in E\}$ . The **parents** of a vertex  $a \in V$  are the vertices  $\text{pa}_G(a) := \{b \in V | b \rightarrow a \in G\}$ , its **neighbors** are the vertices  $\text{ne}_G(a) := \{b \in V | a — b \in G\}$ ; the **degree** of  $a$  is defined as  $\deg(a) := |\{b \in V | (a, b) \in E \vee (b, a) \in E\}|$ , the maximum degree of  $G$  is  $\Delta(G) := \max_{a \in V} \deg(a)$ .

An undirected graph  $G = (V, E)$  is **complete** if all pairs of vertices are neighbors. A **clique** is a subset of vertices  $C \subset V$  such that  $G[C]$  is complete. The **clique number**  $\omega(G)$  of  $G$  is the size of the largest clique in  $G$ .  $G$  is **chordal** if every cycle of length  $k \geq 4$  contains a **chord**, i.e. two nonconsecutive adjacent vertices.

A **directed acyclic graph** or **DAG** is a directed graph without cycles. A partially directed graph  $G = (V, E)$  is a **chain graph** if it contains no *directed* cycle; undirected graphs and DAGs are special cases of chain graphs. Let  $G'$  be the undirected graph we get by removing all directed edges from a chain graph  $G$ . The **chain component**  $T_G(a)$  of a vertex  $a$  is the set of all vertices that are connected to  $a$  in  $G'$ . The set of all chain components of  $G$  is denoted by  $\mathbf{T}(G)$ ; they form a partition of  $V$ . We extend the clique number to chain graphs  $G$  by the definition  $\omega(G) := \max_{T \in \mathbf{T}(G)} \omega(G[T])$ .

An **ordering** of a graph  $G$ , i.e. a permutation  $\sigma : [p] \rightarrow [p]$ , induces a total order on  $V$  by the definition  $a \leq_\sigma b \Leftrightarrow \sigma^{-1}(a) \leq \sigma^{-1}(b)$ . An ordering  $\sigma = (v_1, \dots, v_p)$  is a **perfect elimination ordering** or **PEO** if for all  $i$ ,  $\text{ne}_{G^u} \cap \{v_1, \dots, v_{i-1}\}$  is a clique in  $G^u$ . A **topological ordering** or **TO** of a DAG  $D$  is an

```

Input : $G = ([p], E)$: undirected graph;
 $\sigma = (v_1, \dots, v_p)$: ordering of G .
Output: A proper coloring $c : [p] \rightarrow \mathbb{N}$
 $c(v_1) \leftarrow 1$;
for $i = 2$ to p do
 $c(v_i) \leftarrow \min\{k \in \mathbb{N} \mid k \neq c(u) \forall u \in \{v_1, \dots, v_{i-1}\} \cap \text{ne}(v_i)\}$;
return c ;

```

Algorithm 1: GREEDYCOLORING( $G, \sigma$ ).  
Greedy algorithm that yields a proper coloring of  $G$  along an ordering  $\sigma$ .

ordering  $\sigma$  such that  $a \leq_\sigma b$  for each arrow  $a \rightarrow b \in D$ ; we then say that  $D$  is **oriented according to  $\sigma$** .

For the rest of this section, let  $G = (V, E)$  be an undirected graph. We consider a variant of the breadth-first search (BFS) called **lexicographic BFS** or LEXBFS (Rose, 1970) that takes an ordering  $(v_1, \dots, v_p)$  of  $V$  and the edge set  $E$  as input and that outputs an ordering  $\sigma = \text{LEXBFS}((v_1, \dots, v_p), E)$  listing the vertices of  $V$  in the visited order. If  $\{v_1, \dots, v_k\}$  is a clique,  $\sigma$  also starts with  $v_1, \dots, v_k$ ; we refer to Hauser and Bühlmann (2012) for details of such an implementation. For a set  $A = \{a_1, \dots, a_k\} \subset V$  and an additional vertex  $v \in V \setminus A$ , e.g., we use the notation  $\text{LEXBFS}((A, v, \dots), E)$  to denote a LEXBFS-ordering produced from a start order of the form  $(a_1, \dots, a_k, v, \dots)$ , without specifying the orderings of  $A$  and  $V \setminus (A \cup \{v\})$ .

**Proposition 1** (Rose et al. (1976)). *Let  $G = (V, E)$  be an undirected chordal graph with a LEXBFS-ordering  $\sigma$ . Then  $\sigma$  is also a PEO on  $G$ . By orienting the edges of  $G$  according to  $\sigma$ , we get a DAG without  $v$ -structures.*

Alg. 3 is strongly based on graph colorings. A  **$k$ -coloring** of  $G$  is a map  $c : V \rightarrow [k]$ ; the coloring  $c$  is **proper** if  $c(u) \neq c(v)$  for every edge  $u - v \in G$ . We say that  $G$  is  **$k$ -colorable** if it admits a proper  $k$ -coloring; the **chromatic number**  $\chi(G)$  of  $G$  is the smallest integer  $k$  such that  $G$  is  $k$ -colorable. By greedily coloring the vertices of the graph (see Alg. 1), one gets a proper  $k$ -coloring with  $k \leq \Delta(G) + 1$  in polynomial time (Chvátal, 1984).

For any undirected graph  $G$ , the bounds

$\omega(G) \leq \chi(G) \leq \Delta(G) + 1$  hold.  $G$  is **perfect** if  $\omega(H) = \chi(H)$  holds for every induced subgraph  $H$  of  $G$ . An ordering  $\sigma$  of  $G$  is **perfect** if for any induced subgraph  $H$  of  $G$ , greedy coloring along the ordering induced by  $\sigma$  yields an optimal coloring of  $H$  (i.e., a  $\chi(H)$ -coloring).

**Proposition 2** (Chvátal (1984)). *An ordering  $\sigma$  of an undirected graph  $G$  is perfect if and only if  $G$  has no induced subgraph of the form  $a - b - c - d$  with  $a <_\sigma b$  and  $d <_\sigma c$ .*

It can easily be seen that a PEO fulfills the requirement of Prop. 2; hence we get, together with Prop. 1, the following corollary.

**Corollary 1.** (i) *A perfect elimination ordering on some graph  $G$  is perfect.*

(ii) *Any chordal graph has a perfect ordering.*

**Proposition 3** (Chvátal (1984)). *A graph with a perfect ordering is perfect.*

## 4 Optimal Intervention Targets

An  $\mathcal{I}$ -essential graph is a chain graph with chordal chain components. Their edges are oriented according to a PEO in the DAGs of the corresponding equivalence class; edge orientations of different chain components do not influence (i.e., additionally restrict) each other (Hauser and Bühlmann (2012), Thm. 18 and Prop. 16). We can thus restrict our search for optimal intervention targets to single chain components. We can even treat each chain component as an observational essential graph, as the following lemma shows; we skip a formal proof which is rather simple, but technical.

**Lemma 1.** *Consider an  $\mathcal{I}$ -essential graph  $\mathcal{E}_{\mathcal{I}}(D)$  of some DAG  $D$ , and let  $T \in \mathbf{T}(\mathcal{E}_{\mathcal{I}}(D))$ . Furthermore, let  $I \subset [p]$ ,  $I \notin \mathcal{I}$ , be an (additional) intervention target. Then we have*

$$\mathcal{E}_{\mathcal{I} \cup \{I\}}(D)[T] = \mathcal{E}_{\{\emptyset, I \cap T\}}(D[T]) .$$

### 4.1 Single-vertex Interventions

We start with the treatment of the first active learning approach mentioned in Sect. 2.2. By virtue of the following lemma, the maximum in Eq. (1) can be calculated without enumerating all representative DAGs. The lemma easily follows from Thm. 1; we skip the proof.

**Lemma 2.** Let  $G$  be an  $\mathcal{I}$ -essential graph, and let  $v \in [p]$ . Assume  $D_1$  and  $D_2 \in \mathbf{D}(G)$  such that  $\{a \in \text{ne}_G(v) \mid a \rightarrow v \in D_1\} = \{a \in \text{ne}_G(v) \mid a \rightarrow v \in D_2\} = C$ . Then we have  $D_1 \sim_{\mathcal{I}'} D_2$  under the family of targets  $\mathcal{I}' = \mathcal{I} \cup \{\{v\}\}$ .

The next proposition states that every clique in  $\text{ne}_G(v)$  is an admissible set  $C$  in the sense of Lem. 2 and vice versa. Algorithmically, a DAG  $D$  as described in Prop. 4 can be constructed with LEXBFS; this motivates Alg. 2 which yields a solution of Eq. (1).

**Proposition 4** (Andersson et al. (1997)). Let  $G$  be an undirected chordal graph,  $a \in [p]$  and  $C \subset \text{ne}_G(a)$ . There is a DAG  $D \subset G$  with  $D^u = G$  and  $\{b \in \text{ne}(a) \mid b \rightarrow a \in D\} = C$  which is oriented according to a PEO if and only if  $C$  is a clique.

## 4.2 Interventions at Targets of Arbitrary Size

We now proceed to the solution of Eq. (2). The following proposition, which was already conjectured by Eberhardt (2008), shows that the minimum in Eq. (2) only depends on the clique number of  $G$ :

$$\min_{I' \subset [p]} \max_{D \in \mathbf{D}(G)} \omega(\mathcal{E}_{\mathcal{I} \cup \{I'\}}(D)) = \lceil \omega(G)/2 \rceil.$$

```

Input : $G = ([p], E)$: \mathcal{I} -essential graph.
Output: An optimal intervention vertex $v \in [p]$, or
 \emptyset if G only has directed edges.
 $v_{\text{opt}} \leftarrow 0$; $\eta_{\text{opt}} \leftarrow 0$;
for $v = 1$ to p do
 $\eta_{\text{min}} \leftarrow p^2$;
 foreach clique $C \subset \text{ne}_G(v)$ do
 $\sigma \leftarrow \text{LEXBFS}((C, v, \dots), E[T_G(v)])$;
 $D \leftarrow \text{DAG with skeleton } G, \text{ topological ordering } \sigma$;
 $G' \leftarrow \mathcal{E}_{\{\emptyset, \{v\}\}}(D)$;
 $\eta \leftarrow \text{number of arrows in } G'$;
 if $\eta < \eta_{\text{min}}$ then $\eta_{\text{min}} \leftarrow \eta$;
 if $p^2 > \eta_{\text{min}} > \eta_{\text{opt}}$ then
 $(v_{\text{opt}}, \eta_{\text{opt}}) \leftarrow (v_{\text{opt}}, \eta_{\text{min}})$;
 if $v_{\text{opt}} \neq 0$ then return v_{opt} ;
 else return \emptyset ;

```

Algorithm 2: OPTSINGLE( $G$ ): yields a solution of Eq. (1).

**Proposition 5.** Let  $G$  be an undirected, connected, chordal graph on the vertex set  $V = [p]$ ; such a graph is an observational essential graph. (i) There is an intervention target  $I \subset V$  such that for every DAG  $D \in \mathbf{D}(G)$ , we have

$$\omega(\mathcal{E}_{\{\emptyset, I\}}(D)) \leq \lceil \omega(G)/2 \rceil.$$

(ii) For every intervention target  $I \subset [p]$  there is a DAG  $D \in \mathbf{D}(G)$  such that

$$\omega(\mathcal{E}_{\{\emptyset, I\}}(D)) \geq \lceil \omega(G)/2 \rceil.$$

*Proof.* (i) Since  $G$  is chordal, we have  $\chi(G) = \omega(G)$  by Cor. 1(ii) and Prop. 3. Let  $c : V \rightarrow [\omega(G)]$  be a proper coloring of  $G$ . Define  $I := c^{-1}([h])$  for  $h := \lceil \omega(G)/2 \rceil$ . With an intervention at the target  $I$ , at most the edges of  $G[I]$  and  $G[V \setminus I]$  are unorientable for any causal structure  $D \in \mathbf{D}(G)$  under the family of targets  $\mathcal{I} := \{\emptyset, I\}$ . Therefore the bound

$$\omega(\mathcal{E}_{\mathcal{I}}(D)) \leq \max\{\omega(G[I]), \omega(G[V \setminus I])\}$$

holds for every  $D \in \mathbf{D}(G)$ . It remains to show that both of these terms are bounded by  $h$ .

The induced subgraph  $G[I]$  is also perfect, and  $c|_I$  is a proper  $h$ -coloring of  $G[I]$ . Hence we have  $\omega(G[I]) = \chi(G[I]) \leq h$ . Analogously,  $c|_{V \setminus I}$  is a proper  $(\omega(G) - h)$ -coloring of  $G[V \setminus I]$ , hence we have  $\omega(G[V \setminus I]) = \chi(G[V \setminus I]) \leq \omega(G) - h \leq h$  by definition of  $h$ .

(ii) Let  $C$  be a maximum clique in  $G$ , and define  $C \cap I =: \{v_1, \dots, v_k\}$  and  $C \setminus I =: \{v_{k+1}, \dots, v_\omega\}$ . The LEXBFS-ordering  $\sigma := \text{LEXBFS}((v_1, \dots, v_\omega, \dots), E)$  starts with the vertices  $v_1, \dots, v_\omega$ . Set  $\mathcal{I} := \{\emptyset, I\}$  and let  $D \in \mathbf{D}(G)$  be oriented according to  $\sigma$ .

We claim that the arrows in  $D[C \cap I]$  and in  $D[C \setminus I]$  are not  $\mathcal{I}$ -essential in  $D$ . For  $v_i, v_j \in C \cap I$  ( $i < j$ ), consider the ordering  $\sigma' := \text{LEXBFS}((v_1, \dots, v_j, \dots, v_i, \dots, v_\omega, \dots), E)$ , and  $D' \in \mathbf{D}(G)$  which is obtained by orienting the edges of  $G$  according to  $\sigma'$ . We then have  $D' \sim_{\mathcal{I}} D$ :

- $D$  and  $D'$  obviously have the same skeleton, and both have no v-structures.
- $D^{(I)}$  and  $D'^{(I)}$  have the same skeleton because all arrows between a vertex  $a \in I$  and another one  $b \notin I$  point away from  $a$ .

```

Input : $G = ([p], E)$: essential graph.
Output: An optimal intervention target $I \subset [p]$.
 $I \leftarrow \emptyset$;
foreach $T \in \mathbf{T}(G)$ do
 $\sigma \leftarrow \text{LEXBFS}(T, E[T])$;
 $c \leftarrow \text{GREEDYCOLORING}(G[T], \sigma)$;
 $\omega \leftarrow \max_{v \in [p]} c(v)$; $h \leftarrow \lceil \omega/2 \rceil$;
 $I \leftarrow I \cup c^{-1}([h])$;
return I ;

```

Algorithm 3: OPTUNB( $G$ ): yields a solution of Eq. (2); time complexity is  $O(p + |E|)$ .

For  $v_i, v_j \in C \setminus I$ , the argument is analogous, which proves the claim.

$\mathcal{E}_{\mathcal{I}}(D)$  contains the cliques  $C \cap I$  and  $C \setminus I$  of size  $k$  and  $\omega(G) - k$  though. The fact that  $\max\{k, \omega(G) - k\} \geq \lceil \omega(G)/2 \rceil$  completes the proof.  $\square$

The constructive proof shows that a minimizer  $I$  of Eq. (2) can be generated by means of an optimal coloring which we can get by greedy coloring along a LEXBFS-ordering (see Prop. 1 and Cor. 1); this justifies Alg. 3.

Since an  $\mathcal{I}$ -essential graph has only one representative DAG if and only if its clique number is 1, a direct consequence of Prop. 5 is a (sharp) upper bound on the number of interventions necessary to fully identify a causal model, as it was conjectured by Eberhardt (2008).

**Corollary 2.** *Let  $G$  be the an  $\mathcal{I}$ -essential graph. There is a set of  $k = \lceil \log_2(\omega(G)) \rceil$  intervention targets  $I_1, \dots, I_k$  which are sufficient and in the worst case necessary to make the causal structure fully identifiable:*

$$\mathcal{E}_{\mathcal{I} \cup \{I_1, \dots, I_k\}}(D) = D \quad \forall D \in \mathbf{D}(G).$$

The intervention targets  $I_1, \dots, I_k$  of Cor. 2 can be constructed by iteratively running Alg. 3 on  $G = \mathcal{E}_{\mathcal{I}}(D)$ ,  $\mathcal{E}_{\mathcal{I} \cup \{I_1\}}(D)$ ,  $\mathcal{E}_{\mathcal{I} \cup \{I_1, I_2\}}(D)$  etc. However, they could also be constructed at once by a modification of Alg. 3. Let  $c : [p] \rightarrow [\omega(G)]$  be a function such that for each chain component  $T \in \mathbf{T}(G)$ ,  $c|_T$  is a proper coloring of  $G[T]$ . By defining  $I_j$  as the set of all vertices whose color has a 1 in the  $j^{\text{th}}$  position of its binary representation, we make sure that for every pair of neighboring vertices  $u$  and  $v$ , there

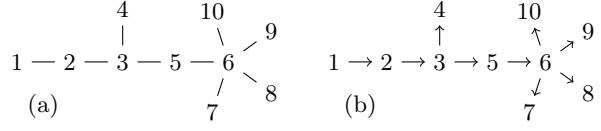


Figure 1: Observational essential graph (a) and a representative (b).

is at least one  $j$  such that  $|\{u, v\} \cap I_j| = 1$ ; hence the edge between  $u$  and  $v$  is orientable in  $\mathcal{E}_{\mathcal{I} \cup \{I_1, \dots, I_k\}}(D)$ . Since the binary representation of  $\omega(G)$ , the largest color in  $c$ , has length  $k = \lceil \log_2(\omega(G)) \rceil$ , this procedure creates a set of  $k$  intervention targets that fulfill the requirements of Cor. 2.

The problem of finding intervention targets to fully identify a causal model is related to the problem of finding separating systems of the chain components of essential graphs (Eberhardt, 2007). A **separating system** of an undirected graph  $G = (V, E)$  is a subset  $\mathcal{F}$  of the powerset of  $V$  such that for each edge  $a - b \in G$ , there is a set  $F \in \mathcal{F}$  with  $|F \cap \{a, b\}| = 1$ . Cai (1984) has shown that the minimum separating system of a graph  $G$  has cardinality  $\lceil \log_2(\chi(G)) \rceil$ ; this also proves Cor. 2. The proof of Cai (1984) uses arguments similar to ours given in the paragraph above for the non-iterative determination of the targets  $I_1, \dots, I_k$  of Cor. 2.

### 4.3 Discussion

LEXBFS and GREEDYCOLORING have a time complexity of  $O(p + |E|)$  when executed on a graph  $G = ([p], E)$ . Thus, OPTUNB (Alg. 3) also has a linear complexity.<sup>1</sup> The time complexity of OPTSINGLE (Alg. 2) on the other hand depends on the size of the largest clique in the  $\mathcal{I}$ -essential graph  $G$ . By restricting OPTSINGLE to  $\mathcal{I}$ -essential graphs with a bounded degree, its complexity is polynomial in  $p$ ; otherwise, it is in the worst case exponential.

We emphasize that our two active learning algorithms do *not* optimize the same objective; OPTUNB does *not* guarantee maximal identifiability after each intervention, and OPTSINGLE

<sup>1</sup>In contrast, finding a minimum separating set on *non-chordal* graphs is NP-complete (Cai, 1984).

does *not* guarantee a minimal number of single-vertex interventions to full identifiability. Consider e.g. the (observational) essential graph in Fig. 1(a). It is not hard to see that all its representatives are fully identifiable after at most two single-vertex interventions: the first intervention should be performed at vertex 3, the second one either at vertex 2 or 6. If the DAG in Fig. 1(b) represents the true causal model, however, OPTSINGLE will need three steps to full identifiability; it will iteratively propose interventions at targets 6, 3 and 2.

In general, OPTUNB does not yield an intervention target of minimal size. With two straightforward improvements, we could reduce the number of intervened vertices: first, we could take  $h \leftarrow \lfloor \omega/2 \rfloor$  instead of  $h \leftarrow \lceil \omega/2 \rceil$  in Alg. 3; the proof of Prop. 5 is also valid with this choice. Second, we could permute the colors produced by the greedy coloring such that  $|c^{-1}(\{1\})| \leq |c^{-1}(\{2\})| \leq \dots$ . However, since an optimal coloring of a graph is not unique, not even up to permutation of colors, these heuristic improvements would still not guarantee a *minimal* intervention target with the properties required in Prop. 5.

## 5 Experimental evaluation

We evaluated Alg. 2 and 3 in a simulation study on 4000 randomly generated causal models with vertex numbers  $p \in \{10, 20, 30, 40\}$ .

### 5.1 Methods

We compared four active learning approaches: our two algorithms OPTSINGLE and OPTUNB, a purely random proposition of single-vertex interventions (denoted by RAND), and a slightly advanced random approach that randomly chooses any vertex which has at least one incident undirected edge (denoted by RANDADV). To measure the quality of the proposed interventions, we evaluated the algorithms together with an “oracle estimator”, that is, an algorithm that yields the true  $\mathcal{I}$ -essential graph of some DAG. This corresponds to model estimation in the limit of infinite sample sizes.

For each vertex number  $p = \{10, 20, 30, 40\}$ , we randomly generated 1000 DAGs with a bi-

nomial distribution of vertex degrees, having an expected degree of 3. Starting from the observational essential graph, we iteratively included the intervention targets proposed by the active learning algorithms. We defined the “survival time” of a DAG as the number of active learning steps needed for full identifiability, measured in intervention targets ( $T$ ) or intervened variables ( $V$ ). If a DAG was fully identifiable e.g. under the family  $\mathcal{I} = \{\emptyset, \{1, 4\}, \{3\}\}$ , we counted  $T = 2$  (non-empty) targets and  $V = 3$  variables. For each vertex number and algorithm, we estimated the “survival function”, i.e. the probability  $S_T(t) := P[T > t]$  or  $S_V(v) := P[V > v]$ , resp., with a Kaplan-Meier estimator (Kaplan and Meier, 1958).

### 5.2 Results

Fig. 2 shows the estimated survival functions of the active learning algorithms. RAND was clearly beaten by all competitors, and OPTUNB dominated all other strategies in terms of intervention targets. However, if we measure the number of intervened vertices, OPTUNB was even slightly worse than RANDADV. OPTSINGLE gave a significant improvement over RANDADV; however, the step from RAND to RANDADV is much larger than from RAND to OPTSINGLE.

When essential graphs are not given by an oracle, but estimated from finite samples with e.g. Greedy Interventional Equivalence Search (Hauser and Bühlmann, 2012), the convergence to the true model is slower due to estimation errors. Performance differences e.g. between RANDADV and OPTSINGLE vanish for small sample sizes (data not shown).

## 6 Conclusion

We developed two algorithms which propose optimal intervention targets: one that finds the single-vertex intervention which maximally increases the number of orientable edges (called OPTSINGLE), and one that maximally reduces the clique number of the non-orientable edges with an intervention at arbitrarily many variables (called OPTUNB). We proved a conjecture of Eberhardt (2008) concerning the number of

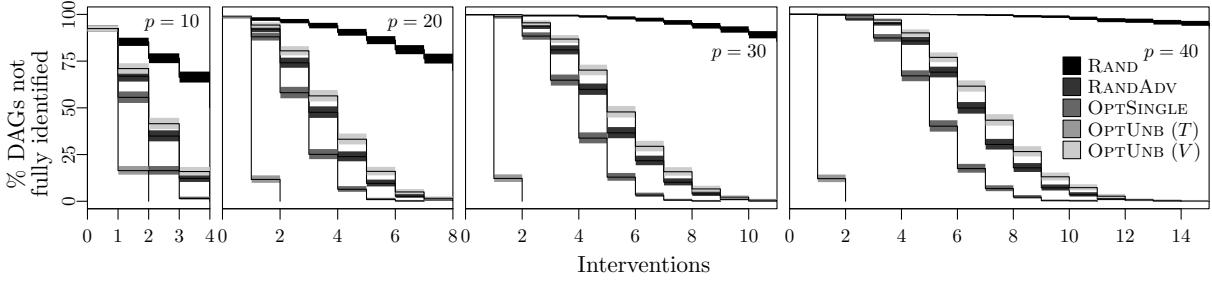


Figure 2: Number of intervention steps needed for full identifiability of DAGs, measured in targets ( $T$ ) or intervened variables ( $V$ ); for algorithms proposing only single-vertex interventions, both numbers are the same. Thin lines: Kaplan-Meier estimates; colored bands: 95% confidence region.

interventions sufficient and in the worst case necessary for fully identifying a causal model by showing that the OPTUNB yields, when applied iteratively, a *minimum* set of intervention targets that guarantee full identifiability.

In a simulation study, we showed that both algorithms lead significantly faster to full identifiability than randomly chosen interventions. If we count the total number of intervened variables, however, OPTUNB performed slightly worse than a random approach. This illustrates the fact that sequentially intervening single variables yields in general more identifiability than intervening those variables simultaneously.

## Acknowledgments

We thank Jonas Peters, Frederick Eberhardt and the anonymous reviewers for valuable comments on the manuscript.

## References

- S.A. Andersson, D. Madigan, and M.D. Perlman. 1997. A characterization of Markov equivalence classes for acyclic digraphs. *Ann. Stat.*, 25(2):505–541.
- M.-C. Cai. 1984. On separating systems of graphs. *Disc. Math.*, 49:15–20.
- V. Chvátal. 1984. Perfectly ordered graphs. *Ann. Disc. Math.*, 21:63–68.
- F. Eberhardt. 2007. *Causation and Intervention*. Ph.D. thesis, Carnegie Mellon University.
- F. Eberhardt. 2008. Almost optimal intervention sets for causal discovery. In *UAI*, pp. 161–168.
- A. Hauser and P. Bühlmann. 2012. Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. *To appear in JMLR; arxiv preprint arXiv:1104.2808*.
- Y.-B. He and Z. Geng. 2008. Active learning of causal networks with intervention experiments and optimal designs. *JMLR*, 9:2523–2547.
- E.L. Kaplan and P. Meier. 1958. Nonparametric estimation from incomplete observations. *JASA*, pp. 457–481.
- K.B. Korb, L.R. Hope, A.E. Nicholson, and K. Axnick. 2004. Varieties of causal intervention. In *PRICAI*, pp. 322–331.
- J. Pearl. 1995. Causal diagrams for empirical research. *Biometrika*, 82:669–688.
- J. Peters, J.M. Mooij, D. Janzing, and B. Schölkopf. 2011. Identifiability of causal graphs using functional models. In *UAI*, pp. 589–598.
- D.J. Rose, R.E. Tarjan, and G.S. Lueker. 1976. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283.
- D.J. Rose. 1970. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597–609.
- P. Spirtes, C.N. Glymour, and R. Scheines. 2000. *Causation, Prediction, and Search*.
- S. Tong and D. Koller. 2001. Active learning for structure in Bayesian networks. In *IJCAI*, vol 17, pp. 863–869.
- T. Verma and J. Pearl. 1990. On the equivalence of causal models. In *UAI*, pp. 220–227.

# Probabilistic Reasoning with Temporal Indeterminacy

Maarten van der Heijden and Peter J.F. Lucas  
Radboud University Nijmegen, The Netherlands  
[{m.vanderheijden,peterl}@cs.ru.nl](mailto:{m.vanderheijden,peterl}@cs.ru.nl)

## Abstract

Temporal indeterminacy, the lack of specific knowledge about the timing of events, occurs often in temporal reasoning in practical applications and is connected to the concept of time granularity. Although logical properties of granularities have been described by several researchers in the literature, the implications of temporal indeterminacy and granularities for probabilistic representation and reasoning have received little attention. Given the widespread occurrence of problems, specifically in medicine, where one has to cope with both temporal indeterminacy and uncertainty, it is somewhat surprising that methods that handle both do not exist as yet. In this paper we propose a formalism to model granularities in temporal Bayesian networks in order to deal with temporal indeterminacy in predictive Bayesian-network models. In addition, some of the properties of multigranular models are explored. Finally, we study a medical use case.

## 1 Introduction

In clinical medicine, as in many other fields, one frequently has to deal with data that is uncertain and records temporal progression. The time aspect is often also uncertain, which is sometimes referred to as *temporal indeterminacy* (Combi and Pozzi, 2001). To build useful predictive models we need to deal with these two kinds of uncertainty. A typical example in clinical medicine would be diagnosis based on a patient report of symptoms, e.g. over the last week. From a modelling perspective it may be uncertain *which* symptoms occurred, and, while our knowledge of *when* the symptoms occurred is constrained to last week, within that period we may not be able to be more specific.

There are many methods that support modelling processes of time; dynamic Bayesian networks (of which hidden Markov models are a special case) are an example (Dean and Kanazawa, 1989; Murphy, 2002). The problem of temporal indeterminacy and its relation to temporal granularities has been recognised in the database community (e.g. Combi and Pozzi (2001)). However, the fundamental problem of dealing with temporal inde-

terminacy in predictive models is mostly unsolved.

The representational power of Bayesian networks makes them a useful tool in a clinical context to represent (causal) relations between symptoms, signs and diseases. We have developed a decision support system for chronic obstructive pulmonary disease (COPD), where patients are monitored at home to detect exacerbation events. In this system a Bayesian network is used for automatic data interpretation (van der Heijden et al., 2011). The dynamic nature of the disease process leads naturally to the desire to employ a model that mirrors the disease and clinical practice more closely than a static Bayesian network. The first step is then to extend the Bayesian network to a dynamic Bayesian network (DBN) which allows us to model dependences through time.

Temporal indeterminacy results from the harsh reality of patient monitoring in a home environment, in the sense that it may not always be possible to obtain precise information at the right time. We want to be capable of dealing with indeterminacy of the kind “symptom  $S$  appeared between 2 and 4 days ago”. This problem is not limited to monitoring, but ap-

pears often in clinical practice. We can identify similar aggregation problems: when lots of data is available (e.g. on the intensive care unit), aggregation as a summary is sometimes useful; a situation that occurs frequently is that measurements are taken at different times but the clinician is interested in an aggregated result that represents the health status of the patient over the period the measurements were taken. These problems have been studied in the context of temporal abstraction (Shahar, 1997), but not in probabilistic graphical models.

We aim to develop a framework to specify temporal indeterminacy, and related aggregation problems, probabilistically. In this paper we focus on this representational problem in the context of dynamic Bayesian networks, with the objective to make predictive models that handle indeterminacy.

## 2 Preliminaries

### 2.1 Representing time

In order to reason about temporal indeterminacy we need a formalism to represent time and granularity. The work on representing time in temporal databases seems useful as a starting point (Combi and Pozzi, 2001). We take a linearly ordered set of points ( $\mathcal{T}, \leq$ ) as the primitive representation of a time line, where  $\mathcal{T}$  is a subset of the natural numbers  $\mathbb{N}$ . A *determined instant* at the lowest time granularity (e.g. seconds) is a point  $t \in \mathcal{T}$ . Due to temporal indeterminacy and granularities, it may be useful to represent an instant as a set. For example, an instant at the scale of minutes is a single minute, but may be represented as a set of 60 seconds at a finer granularity. Similarly, when the exact time of an instant is unknown, a set of points can be used to represent the uncertainty in time, that is, to represent the temporal indeterminacy. An *indeterminate instant* is defined as a set  $a \subseteq \mathcal{T}$ , with the property that  $a$  is contiguous:  $\forall t \in \mathcal{T} : \inf a \leq t \leq \sup a \Leftrightarrow t \in a$ , using  $\inf a$  and  $\sup a$  to denote the lower and upper bound of the interval of indeterminacy. Two instants  $a, b$  are called non-overlapping if  $\sup a < \inf b$  or  $\sup b < \inf a$ .

In studies on logical representations of granularities (e.g. Bettini et al. (1998)), different time structures are allowed as granularities. This allows one to specify hierarchies of granularities for example for the Gregorian calendar. Here we define a granularity  $G$  in a more restricted sense as a partition of  $\mathcal{T}$ , i.e. a set of subsets such that  $\bigcup G = \mathcal{T}$  and if  $g, g' \in G$  then  $g \cap g' = \emptyset$  or  $g = g'$ . We further consider only granularities that are contiguous, uniform and comparable, which means that every time point can be expressed in each granularity and time points within a granularity have the same size (which excludes for example ‘month’). Examples of possible granularities with these restrictions include ‘second’, ‘hour’, ‘day’ etc.

Before we go on we introduce some notation. We are often interested in events associated with time. In particular we consider events of observing a value of a certain variable of interest. We write  $E_a$  for an event  $E$  that occurs in  $a$ , where  $a$  is an indeterminate instant. It is sometimes convenient to specify a certain time point  $t$  within  $a$ . For example, to denote the probability that  $E$  occurs at  $t \in a$ , with  $a$  an instant, we write  $P(E_a(t))$ . The complement event of  $E_a$  is denoted  $\bar{E}_a$ .

### 2.2 Bayesian networks

To be able to add temporal indeterminacy to temporal Bayesian networks, we first define the usual way to explicitly incorporate time in Bayesian networks.

A *dynamic Bayesian network* is a pair  $(G, \mathbf{F})$ , with  $G$  a graph  $G = (\mathbf{V}, \mathbf{A})$  and  $\mathbf{F}$  a set of factors over random variables  $\mathbf{X}$  corresponding to the vertices  $\mathbf{V}$ . The set of arcs in the graph  $\mathbf{A} \subseteq \mathbf{V} \times \mathbf{V}$  represents (in)dependences of the variables. For a two-slice dynamic Bayesian network we subdivide the vertices in an initial slice and a repeated transition slice  $\mathbf{V} = \mathbf{V}_0 \cup \mathbf{V}_{1:T}$ , which implies a similar subdivision  $\mathbf{X} = \mathbf{X}_0 \cup \mathbf{X}_{1:T}$ . An arc  $(v, w) \in \mathbf{A}$  with  $v \in \mathbf{V}_t$  and  $w \in \mathbf{V}_{t'}, t < t'$ , denotes a temporal dependence. Let  $pa(X)$  denote the parents of  $X \in \mathbf{X}$  in the graph  $G$ . The set  $\mathbf{F}$  contains factors for each  $X \in \mathbf{X}$  such that  $f(x, \mathbf{p}) = P(X = x | pa(X) = \mathbf{p})$ . The joint

distribution over  $\mathbf{X} = \bigcup_i^T \mathbf{X}_i$ , factorises over the graph such that  $P(\mathbf{X}) = \prod_{X \in \mathbf{X}} P(X | pa(X))$ .

To simplify the models, often only first-order Markov models are considered, which means that the future is independent of the past given the present:

$$P(X_{t+1} | X_t, X_{1:t-1}) = P(X_{t+1} | X_t).$$

Furthermore, a usual assumption is homogeneity, also called stationarity, which means that the probabilities are equal between time steps:

$$\begin{aligned} P(X_t | pa(X_t)) &= P(X_{t'} | pa(X_{t'})) \\ \forall t, t' : X_t \in \mathbf{X}_t, X_{t'} \in \mathbf{X}_{t'} \end{aligned}$$

This may be too strong an assumption for real processes, which has lead to recent work on learning non-stationary dynamic Bayesian networks (Robinson and Hartemink, 2010; Grzegorczyk and Husmeier, 2011). We leave these assumptions intact here, and focus only on the indeterminacy problem.

A technique often used to model the interaction between multiple variables (to prevent an exponential growth of parameters) are *causal independence* models (Pearl, 1988). The main assumption is that different causes of an effect can be assumed to be independent and only interact through a particular deterministic function. Here we will use this method to model the interaction between random variables at different time granularities.

### 3 Events and granularity

We first state some properties of indeterminate events, and their consequences. Consider an event  $E_a$  to be a point-like occurrence at the finest granularity under consideration somewhere at  $a$ . We then have:

$$E_a = \bigvee_{t \in a} E_a(t), \quad (1)$$

i.e., event  $E_a$  is defined in terms of events at the finer granularity. Analogously, the complement event is:

$$\bar{E}_a = \neg \bigvee_{t \in a} E_a(t) = \bigwedge_{t \in a} \bar{E}_a(t). \quad (2)$$

The following property ensures that only a single point event  $E_a(t)$  is true:

$$\forall t, t' \in a : t \neq t' \rightarrow E_a(t) \wedge E_a(t') = \perp. \quad (3)$$

called the *single event assumption*. This means that the event at coarser granularity is caused by exactly one event at finer granularity. Conversely, if multiple events could have occurred at the finer granularity, this is called the *multiple event assumption*.

When using probability theory to capture the principles of an indeterminate instant  $a$ , we need to specify a distribution for the occurrence of an event  $E$  associated with the instant  $a$ . Under the single event assumption a distribution for  $E_a$  has the property:

$$P(E_a) = \sum_{t \in a} P(E_a(t)),$$

which follows from the two properties above. From Property (3) it also follows directly that:

$$\begin{aligned} \forall t, t' \in a : t \neq t' \rightarrow \\ P(\dots, E_a(t), \dots, E_a(t'), \dots) = 0 \end{aligned} \quad (4)$$

and

$$P\left(\bigwedge_{t \in a \setminus \{t'\}} \bar{E}_a(t) \mid E_a(t')\right) = 1.$$

From this, it follows that

$$\begin{aligned} P\left(\bigwedge_{t \in a \setminus \{t'\}} \bar{E}_a(t), E_a(t')\right) \\ = P\left(\bigwedge_{t \in a \setminus \{t'\}} \bar{E}_a(t) \mid E_a(t')\right) P(E_a(t')) \\ = 1 \cdot P(E_a(t')) = P(E_a(t')). \end{aligned} \quad (5)$$

However, when the multiple event assumption is adopted, only

$$P(E_a) = P\left(\bigvee_{t \in a} E_a(t)\right)$$

holds. This means that at the aggregated level we cannot distinguish between what are multiple events at the precise granularity. We are interested in somehow relating the representations at both granularities.

Further constraints result from domain specific patterns of dependence between points in  $a$ , which will be considered later.

## 4 Temporal indeterminacy in DBNs

Now we focus on modelling temporal indeterminacy in dynamic Bayesian networks. In general the problem is representing a temporal process at multiple granularities.

### 4.1 Probabilistic aggregation

A conceptually simple representation consists of separate models for the granularities. This is shown in Figure 1. Basically, the lower part of the model aggregates the top part, although here it is not specified how the aggregation takes place. A one-to-one translation from the possible states of the finer granularity to the coarser granularity results in:

$$P(O_a | X_a) P(X_a) = P\left(\bigwedge_{i=1}^n O_i \mid \bigwedge_{j=1}^n X_j\right) P\left(\bigwedge_{k=1}^n X_k\right),$$

where basically the domain of  $O_a$  is the Cartesian product of the domains of  $O_1, \dots, O_n$ ; similarly, the domain of  $X_a$  is the Cartesian product of the domains of  $X_1, \dots, X_n$ . The result is a representation that is exponential in the size of the domains.

With the single event assumption it is easily possible to compute the aggregated probabilities. However, when represented graphically, these logical constraints would require dropping the first-order Markov assumption yielding a factorisation of  $P(X_1, \dots, X_n)$  without any independence information. However, in the transformation it is not necessary to use a graphical representation as there is a very specific relation between the granularities. Let  $X_a$  take on values in  $\{0, 1, \dots, n\}$  then  $P(X_a = i) = P(X_i = 1)$  and  $P(X_a = 0) = 1 - \sum_{i=1}^n P(X_a = i)$ , because by definition for  $i$  with  $1 \leq i \leq n$ :

$$\begin{aligned} P(X_a = i) &= P(X_1 = 0, \dots, X_i = 1, \dots, X_n = 0) \\ &= P(X_n = 0, \dots, X_{i+1} = 0 | X_i = 1, \dots, X_1 = 0) \\ &\quad \cdot P(X_i = 1, X_{i-1} = 0, \dots, X_1 = 0) \\ &= 1 \cdot P(X_i = 1, X_{i-1} = 0, \dots, X_1 = 0) \\ &= P(X_i = 1) \end{aligned}$$

using Equation (5). Note that it follows that

$$P(X_a \neq i) = P(X_i = 0).$$

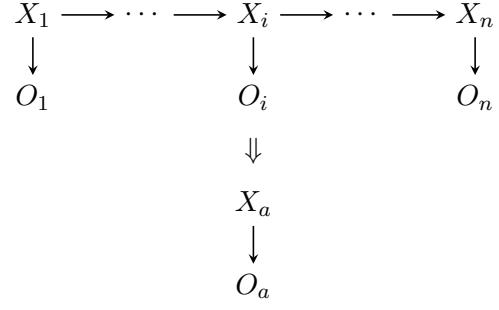


Figure 1: A general depiction of a model at two granularities (without the single event assumption).

$$\begin{aligned} P(X_i = 1) &= \sum_{X_j, i \neq j} P(X_i = 1, \bigwedge_j X_j) \\ &= P(X_i = 1, \bigwedge_j X_j = 0), \end{aligned}$$

where the second equality follows from mutual exclusivity, as is easily seen from Equation (4). Since this is true for  $X_i$  with  $1 \leq i \leq n$  and  $P(X_a)$  is normalised by setting  $P(X_a = 0) = 1 - \sum_{i=1}^n P(X_a = i)$ , we obtain the aggregation  $P(X_a) = P(\bigwedge_{i=1}^n X_i)$ .

To aggregate the observation variables  $O_i$  we take the domain of  $O_a$  to be binary. Since we assumed stationarity we have only two parameters: for  $1 \leq i \leq n$   $P(O_i = 1 | X_i = 1) = p$  and  $P(O_i = 1 | X_i = 0) = q$ . Still assuming that the first-order Markov assumption does not hold, conditioned on the  $X_i$ 's the  $O_i$  variables are conditionally independent, allowing us to aggregate the observations by multiplying the conditional probabilities; for  $1 \leq i \leq n$ , it holds that:

$$P(O_a = 1 | X_a = i) = P(O_i = 1 | X_i = 1) \cdot \prod_{j \neq i} P(O_j = 1 | X_j = 0) = pq^{n-1}$$

and in addition

$$\begin{aligned} P(O_a = 1 | X_a = 0) &= \prod_j P(O_j = 1 | X_j = 0) = q^n. \end{aligned}$$

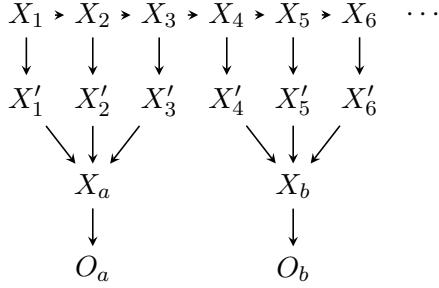


Figure 2: Incorporating granularity within the model. Observation variables for the finer granularity have been omitted.

## 4.2 Structural aggregation

A different perspective on modelling indeterminacy is including the aggregate temporal structure within the model. This turns out to be useful as it allows us to model explicitly the interaction between the variables at different granularities. The mechanism that we will employ to model multiple interactions in a systematic way is *causal independence*. This is depicted in Figure 2. A causal independence model allows us to model the projection by means of a particular deterministic function of the variables at the finer granularity.

Although separating the state and observation model is common practice to model measurement errors, we will for now assume that state variables are directly observable in order to focus on modelling the indeterminacy. Hence, disregarding the  $O$  variables in Figure 2 we obtain the following joint distribution for the aggregation over  $a$

$$P(X_a, \bigwedge_{i=1}^n X'_i, X_i) = P(X_a | \bigwedge_{i=1}^n X'_i) \prod_j P(X'_j | X_j) \\ P(X_1) \cdots P(X_i | X_{i-1}) \cdots P(X_n | X_{n-1}). \quad (6)$$

In a causal independence model the term  $P(X_a | \bigwedge_{i=1}^n X'_i)$  is represented by a deterministic function. Equation (1) implies that the logical OR is a natural interaction function.

As the single event assumption is easily represented by the construction in the previous section, we now focus on the multiple event assumption. The causal independence model al-

lows us to combine the events, even when the independence assumption on the causes is not met (as can be seen to be the case in Figure 2). Although the model complexity increases, the number of parameters is still linear in the number of variables at the fine granularity, as opposed to the exponential increase which results when connecting the granularities directly.

The model depicted in Figure 2 is not the only possible aggregation. An aggregation variable can summarise arbitrary sets of state variables, which is related to the work by Evers et al. (2008). The statement “symptom  $S$  occurred between 2 and 4 days ago” can be modelled with an aggregation variable that summarises 3 variables at the granularity of days. These aggregations can be made to overlap, that is  $X_a$  summarises  $X_{1:3}$ ,  $X_b$  summarises  $X_{2:4}$  etcetera.

## Aggregation patterns

Starting from the aggregated level we can use domain knowledge to specify a pattern on the finer granularity. It seems worthwhile to identify patterns that have intuitive appeal, defining which variables have more influence on the aggregated variable, or, from the perspective of the coarser granularity, which variables are more likely to have caused the aggregated state. Specifically, we consider four situations: the *uniformity* assumption implies that we have no reason to believe that there is a particular structure at the precise granularity, the maximum entropy choice. A *discretised normal* distribution is useful to model the uncertainty around a particular observation point. The *decreasing* and *increasing* patterns convey the idea that some event is more likely to occur at the start or the end of the aggregated interval.

If a point in the coarser granularity  $X_a$  has corresponding points in the finer granularity  $X_1, \dots, X_i, \dots, X_n$ , the patterns have the following properties:

Uniform:  $P(X_1) = P(X_i) = P(X_n)$

Normal:  $P(X_i) = P(X_{n-i+1}) < P(X_{n/2})$

Decreasing:  $P(X_1) \geq P(X_i) \geq P(X_n)$

Increasing:  $P(X_1) \leq P(X_i) \leq P(X_n)$

Note that for the *normal*-pattern some care needs to be taken when aggregating an even number of variables. For the *increasing* and *decreasing* patterns one could also consider the strict cases, but especially for large  $n$  the current versions are more flexible.

## 5 Temporal indeterminacy in COPD

The model shown in Figure 3 is an example of how temporal aggregation can be used in the context of COPD-monitoring. The data has been gathered at various hospitals and general practices in the Netherlands, with the smartphone-based monitoring system as described in (van der Heijden et al., 2011). The model is an unrolled dynamic Bayesian network for four symptoms: dyspnea, cough, sputum production and activity capacity. Each variable indicates a daily measurement over the course of a week, and takes on binary values (False = normal, True = worse than normal). The daily symptom variables are aggregated in a variable representing the whole week by means of a generalised causal independence model (Srinivas, 1993). Instead of a binary outcome variable, the aggregate has values representing the number of symptom days (0 through 8). The intermediate variables model the inhibition, or noise, parameters. Finally, we are interested in the outcome variable *Exacerbation*, which provides an indication of a clinical significant event. Here again a causal independence model is utilised to aggregate the individual symptoms, by means of a counting function that classifies the number of symptoms into the categories [0-7],[8-15],[16-23],[24-32], that is, it is a deterministic function of the symptom aggregates.

The parameters of the model have been learned from the monitoring data of 8 COPD patients that participated in a pilot study. As before we assumed stationarity, and with a Dirichlet uninformative prior distribution with  $\alpha = 1.1$  we estimated the probabilities of a symptom variable  $X \in \{D, C, S, A\}$  as:

$$P(X_i = k | pa(X_i) = j) = \frac{\alpha_{ijk} + N_{ijk} - 1}{\alpha_{ij} + N_{ij} - |X_i|},$$

where  $N_{ijk}$  is the number of cases that variable  $i$  takes value  $k$  and the parent variables of  $i$  take on configuration  $j$ ;  $\alpha_{ijk}$  denotes the pseudo-counts;  $|X_i|$  is the number of values that variable  $X_i$  can take on and  $N_{ij} = \sum_k N_{ijk}$ . The parameters of the intermediate variables were all equal with  $P(XIn_i = 1 | X_i = 1) = 0.9$  and  $P(XIn_i = 1 | X_i = 0) = 0.1$ . All the aggregate variables have deterministic parameters as explained above.

In the pilot study, participants were asked to fill in the Clinical COPD Questionnaire (CCQ) as a control measure for the data gathered with the monitoring system. The questionnaire consists of 10 questions on a 7 point scale (lower scores are better); 4 questions are about dyspnea, 1 about cough, 1 about sputum production and 4 about activity; we omitted the 2 psychological questions about dyspnea for the current analysis; the version used asked patients to answer with respect to last week. For our current purposes this gives us a good example to look at aggregation, since we can compare the monitoring data from the preceding week with the results of the CCQ. Because not all data was available at the time of writing and due to missing data (patients not filling in the CCQ), we have data of 5 patients for comparison.

The procedure is as follows, we entered the monitoring data as evidence in the DBN (missing values are easily taken care of by leaving them as non-evidence variables) and noted the probabilities of the aggregate variables. For the CCQ scores we took the mean of the scores for the same aspect, rounded to the nearest integer. The results of the comparison are shown in Table 1, where the distribution over the aggregate variables is shown, which basically means the probability of a certain number of days of that particular symptom during the preceding week. Each value of an aggregate variable is mapped to the same CCQ score (0 to 6), except 7 which is mapped to CCQ score 6 (i.e. ‘(almost) always’ is taken to mean 6 or 7 days).

Using the most likely value of the variables as a simple measure to assess agreement with the CCQ scores we see that we end up with the same value in 30% of the cases and are one value

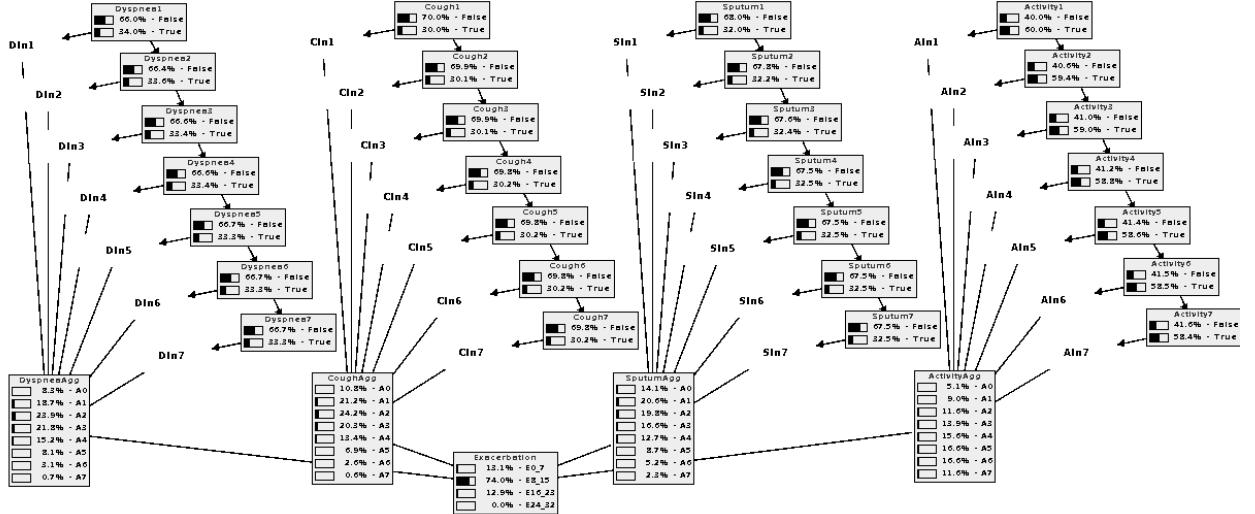


Figure 3: COPD-monitoring aggregation network.

removed in 45%. But given the currently limited amount of data it is hard to say whether this could be improved by a better aggregation model or whether there exists a discrepancy between the monitoring data and the CCQ scores. The sputum value of patient 3 favours the latter explanation, as the CCQ indicates sputum was produced almost never whereas the monitoring data provide evidence to the contrary; clearly an aggregation model cannot solve these inconsistencies. With respect to the outcome variable *Exacerbation*, visual inspection of Table 1 indicates that patients 1 and 2 are stable, patients 3 and 5 are in the course of an exacerbation and patient 4 is somewhere in between. This shows that the aggregation provides a useful and more easily interpreted summary of the monitoring data.

## 6 Related work

A well known representation of time is Allen's algebra (Allen, 1983). It provides a set of relations between time intervals and operations to reason over the intervals. The relations allow expressing information about the order of intervals and some different types of overlap, for example 'X starts or is during Y'. In some sense this allows indeterminacy, but in a qualitative way. Temporal indeterminacy and its

relation to representations at different granularities is a topic studied in the context of temporal databases (Combi and Pozzi, 2001); but the problem is also relevant for planning and scheduling and information systems that deal with temporal data. For medical information systems there has been quite some work on temporal reasoning: on multigranular representations (Keravnou, 1999); and on temporal abstraction (Shahar, 1997). The latter deals with aggregating temporal data to meaningful higher level concepts, which requires taking care of temporal representation issues like granularities. Our probabilistic graphical model approach provides a different view on a part of these problems. Also related are irregular-time Bayesian networks (Ramati and Shahar, 2010), which generalise DBNs to time-slices with changing size. Bettini et al. (1998) studied granularities in temporal constraint satisfaction problems; and Combi et al. (2004) in a more general linear time logic context, both deal extensively with properties of multiple granularities but not with probabilistic representations.

## 7 Conclusion

Modelling temporal indeterminacy in probabilistic graphical models creates the opportunity to deal with different kinds of uncertainty

Table 1: Distributions (percentage) over the aggregate variables (D=dyspnea, C=cough, S=sputum, A=activity) for 5 patients. In bold the probability of the value that corresponds to the CCQ score.

|   | 0           | 1           | 2           | 3           | 4           | 5           | 6           | 7          |
|---|-------------|-------------|-------------|-------------|-------------|-------------|-------------|------------|
| D | 47.8        | 37.2        | <b>12.4</b> | 2.3         | 0.3         | 0           | 0           | 0          |
| C | 0.6         | <b>11.0</b> | 53.8        | 27.9        | 6.1         | 0.7         | 0           | 0          |
| S | <b>4.7</b>  | 45.7        | 35.8        | 11.6        | 2.0         | 0.2         | 0           | 0          |
| A | <b>47.8</b> | 37.2        | 12.4        | 2.3         | 0.3         | 0           | 0           | 0          |
|   |             |             |             |             |             |             |             |            |
| D | 30.6        | <b>37.8</b> | 23.0        | 7.2         | 1.3         | 0.1         | 0           | 0          |
| C | 32.7        | <b>37.8</b> | 21.7        | 6.6         | 1.1         | 0.1         | 0           | 0          |
| S | 36.3        | 36.9        | <b>19.8</b> | 5.9         | 1.0         | 0.1         | 0           | 0          |
| A | <b>31.0</b> | 35.3        | 23.8        | 8.2         | 1.5         | 0.2         | 0           | 0          |
|   |             |             |             |             |             |             |             |            |
| D | 0           | 0.5         | 4.8         | 20.7        | 33.5        | <b>28.5</b> | 11.1        | 0.9        |
| C | 0           | 0           | 0.5         | 4.3         | <b>17.5</b> | 34.0        | 29.9        | 13.7       |
| S | 0           | <b>0</b>    | 0.3         | 2.4         | 11.5        | 28.4        | 33.7        | 23.6       |
| A | 0           | 0           | 0.1         | 1.2         | 6.7         | <b>20.8</b> | 36.6        | 34.7       |
|   |             |             |             |             |             |             |             |            |
| D | 0           | 1.0         | 7.8         | 23.4        | <b>32.9</b> | 25.1        | 9.0         | 0.7        |
| C | 1.2         | 13.8        | 29.7        | <b>32.0</b> | 18.1        | 4.7         | 0.5         | 0          |
| S | 33.2        | 37.1        | <b>21.2</b> | 7.0         | 1.3         | 0.1         | 0           | 0          |
| A | 0           | 0.2         | 1.9         | <b>9.6</b>  | 24.6        | 37.8        | 23.7        | 2.2        |
|   |             |             |             |             |             |             |             |            |
| D | 0           | 0           | 0.2         | 2.4         | 13.5        | 37.8        | <b>41.8</b> | <b>4.2</b> |
| C | 0           | 1.0         | <b>9.7</b>  | 36.8        | 37.3        | 13.3        | 1.8         | 0.1        |
| S | 0           | 0           | 0.4         | 3.9         | 19.4        | 44.3        | <b>29.2</b> | <b>2.7</b> |
| A | 0           | 0           | 0           | 0.5         | 3.7         | <b>15.5</b> | 37.4        | 42.8       |

that arises in many practical situations and in chronic disease monitoring in particular. We think that our initial results on probabilistic multigranular models by means of causal independence form a good starting point for further research in this direction. Particularly, the consequences and applicability of the single or multiple event assumption warrant attention. Future work also includes further analysis in the context of COPD-monitoring.

## References

- J. Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.
- C. Bettini, X. Wang, and S. Jajodia. 1998. A general framework for time granularity and its application to temporal reasoning. *Annals of Mathematics and Artificial Intelligence*, 22:29–58.
- C. Combi and G. Pozzi. 2001. HMAP – A temporal data model managing intervals with different granularities and indeterminacy from natural language sentences. *The VLDB Journal*, 9(4):294–311.
- C. Combi, M. Franceschet, and A. Peron. 2004. Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, 14(1):51–77.
- T. Dean and K. Kanazawa. 1989. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150.
- S. Evers, M. Fokkinga, and P. Apers. 2008. Probabilistic processing of interval-valued sensor data. In *Proceedings of the 5th workshop on Data management for sensor networks*.
- M. Grzegorczyk and D. Husmeier. 2011. Non-homogeneous dynamic Bayesian networks for continuous data. *Machine Learning*, 83(3):355–419.
- E. Keravnou. 1999. A multidimensional and multi-granular model of time for medical knowledge-based systems. *Journal of Intelligent Information Systems*, 13:73–120.
- Kevin P. Murphy. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California, Berkeley.
- Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- M. Ramati and Y. Shahar. 2010. Irregular-time Bayesian networks. In *UAI ’10: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*.
- J. Robinson and A. Hartemink. 2010. Learning non-stationary dynamic Bayesian networks. *Journal of Machine Learning Research*, 11:3647–3680.
- Y. Shahar. 1997. A framework for knowledge-based temporal abstraction. *Artificial Intelligence*, 90(1-2):79–133.
- S. Srinivas. 1993. A generalization of the noisy-or model. In *UAI ’93: Proceedings of the 9th conference on Uncertainty in Artificial Intelligence*.
- M. van der Heijden, B. Lijnse, P.J.F. Lucas, Y.F. Heijdra, and T.R.J. Schermer. 2011. Managing COPD exacerbations with telemedicine. In M. Peleg, N. Lavrač, and C. Combi, editors, *AIME ’11: Proceedings of the 13th Conference on Artificial Intelligence in Medicine*, volume 6747 of *LNAI*, pages 169–178.

# Hybrid Binary–Chain Multi-label Classifiers

Pablo Hernandez-Leal, Felipe Orihuela-Espina, L. Enrique Sucar and Eduardo F. Morales

Instituto Nacional de Astrofísica, Óptica y Electrónica

Luis Enrique Erro No. 1, Sta. Ma. Tonantzintla, Puebla, Mexico

{pablol,f.orihuela-espina,esucar,emorales}@inaoep.mx

## Abstract

In multi-label classification the goal is to assign an instance to a set of different classes. Several approaches have been proposed to deal with multi-label classification problems, ranging from considering each class independently from the other (binary relevance methods) to considering all the possible combinations of values of the original classes into a single compound class (power-set approach). In between, other methods have been proposed to consider dependencies among classes whilst trying to keep computational complexity of the method low. In this paper, instead of finding probabilistic dependencies among classes, we focused on finding independencies among classes using a simple correlation approach. We first build a correlation matrix among classes and use it to build chain classifiers among correlated sub-sets of classes while learning independent classifiers for uncorrelated classes. It is experimentally shown that this simple hybrid approach exhibits very competitive predictive performance among state-of-the-art multi-label classifiers with lower time complexity.

## 1 Introduction

Multi-label classifiers (MLCs) have gained an increasing attention in recent years, as different important problems can be seen as multi-label classification (Zhang and Zhou, 2007; Vens et al., 2008), such as text classification (assigning a document to several topics), HIV drug selection (determining the optimal set of drugs), among others.

There are basically two types of approaches that have been proposed for solving an MLC problem with binary classes: *binary relevance* and *label power-set* (Tsoumakas and Katakis, 2007). In the binary relevance approach (Zhang and Zhou, 2007), an MLC problem is transformed into  $d$  binary classification problems, one for each class variable,  $C_1, \dots, C_d$ . A classifier is independently learned for each class variable, and the results are combined to determine the predicted class set. The problem with this approach is that it is unable to capture the interactions among classes and, in general, the most likely class of each classifier will not match

the most likely set of classes due to possible interactions among them.

The label power-set approach (Tsoumakas and Katakis, 2007) on the other hand transforms the multi-label problem into a single-class version by defining a new class variable whose possible values are all of the possible combinations of values of the original classes. In this case the interactions between the different classes are implicitly considered. One disadvantage of this approach is its computational complexity, as the size of the new class variable increases exponentially with the number of classes.

Recently, alternative approaches have been proposed to consider dependencies among classes without incurring in high computational costs. One of these approaches is chain classifiers (Read et al., 2011), that consist of  $d$  binary classifiers which are linked in a chain, such that each classifier incorporates the class predicted by the previous classifiers as additional attributes. This approach combines the computational advantages of binary-relevance meth-

ods while incorporating dependencies among classes considered in the chain. The chain order, however, is randomly selected and does not necessarily capture the actual dependencies among classes while incorporating much irrelevant information from independent classes. A related approach is presented in (Zaragoza et al., 2011), where the authors use a probabilistic approach to establish a dependency class structure to build chain classifiers.

In this work, instead of looking for dependencies among classes, we look for independencies between classes using a simple correlation analysis. We first build a matrix of pairwise correlations between classes. Since uncorrelated classes can be considered independent, we build independent classifiers for them. For each subset of correlated classes, we group them and induce a simple chain classifier among them. We show, experimentally, that by identifying independent classes, we are able to significantly reduce the computation time while achieving competitive predictive performance results against state-of-the art multi-label classifiers.

## 2 Multi-Label Classifiers

The *multi-dimensional classification* problem corresponds to searching for a function  $h$  that assigns to each instance represented by a vector of  $m$  features  $\mathbf{x} = (x_1, \dots, x_m)$  a vector of  $d$  class values  $\mathbf{c} = (c_1, \dots, c_d)$ :

$$h : \Omega_{X_1} \times \dots \times \Omega_{X_m} \rightarrow \Omega_{C_1} \times \dots \times \Omega_{C_d}$$

$$(x_1, \dots, x_m) \mapsto (c_1, \dots, c_d)$$

where there are  $d$  class variables,  $C_1, \dots, C_d$ . We assume that  $C_{i|i=1, \dots, d}$  and  $X_{j|j=1, \dots, m}$  are discrete, and that  $\Omega_{C_i}$  and  $\Omega_{X_j}$  respectively represent their sample spaces.

Under a  $0 - 1$  loss function, the  $h$  function should assign to each instance  $\mathbf{x}$  the most likely combination of classes, that is:

$$h(x) = \operatorname{argmax}_{c_1, \dots, c_d} p(C_1 = c_1, \dots, C_d = c_d | \mathbf{x})$$

This assignment amounts to solving a total abduction inference problem and corresponds

to the search for the most probable explanation (MPE), a problem that has been proved to be an NP-hard problem for Bayesian networks (Shimony, 1994).

## 3 Related Work

### 3.1 Multi-label Classification

In multi-label classification domains each instance is associated with a subset of labels from a set of  $d$  labels. This multi-label classification problem can be seen as a particular case of a multidimensional classification problem where all class variables are binary, that is  $|\Omega_{C_i}| = 2$  for  $i = 1, \dots, d$ .

An overview of multi-label classification is given in (Tsoumakas and Katakis, 2007). Two main approaches are distinguished: (a) problem transformation methods, which transform the multi-label classification problem into either one or more single-label classification problems and (b) algorithm adaptation methods, which extend specific learning algorithms to handle multi-label data directly.

Other related approaches are multidimensional Bayesian network classifiers (MBCs). A MBC is a Bayesian network  $B = (\mathcal{G}, \Theta)$ , where  $\mathcal{G}$  is an acyclic directed graph with vertexes  $Z_i$  and  $\Theta$  is a set of parameters  $\theta_{z|\mathbf{pa}(z)} = p(z|\mathbf{pa}(z))$ , where  $\mathbf{pa}(z)$  is a value for the set  $\mathbf{Pa}(Z)$ , parents variables of  $Z$  in  $\mathcal{G}$ .

The set of vertexes  $\mathcal{V}$  is partitioned into two sets  $\mathcal{V}_C = \{C_1, \dots, C_d\}$ ,  $d \geq 1$ , of class variables and  $\mathcal{V}_X = \{X_1, \dots, X_m\}$ ,  $m \geq 1$ , of feature variables. The set  $\mathcal{A}$  of arcs is also partitioned into three sets,  $\mathcal{A}_C$ ,  $\mathcal{A}_X$ ,  $\mathcal{A}_{CX}$ , such that  $\mathcal{A}_C \subseteq \mathcal{V}_C \times \mathcal{V}_C$  is composed of the arcs between the class variables,  $\mathcal{A}_X \subseteq \mathcal{V}_X \times \mathcal{V}_X$  is composed of the arcs between the feature variables and finally,  $\mathcal{A}_{CX} \subseteq \mathcal{V}_C \times \mathcal{V}_X$  is composed of the arcs from the class variables to the feature variables. The corresponding induced subgraphs are  $\mathcal{G}_C = (\mathcal{V}_C, \mathcal{A}_C)$ ,  $\mathcal{G}_X = (\mathcal{V}_X, \mathcal{A}_X)$  and  $\mathcal{G}_{CX} = (\mathcal{V}, \mathcal{A}_{CX})$ , called respectively class, feature and bridge subgraphs.

Different graphical structures for the class and feature subgraphs lead to different families of MBCs. For instance, a simple approach is to

learn trees for both subgraphs (van der Gaag and de Waal, 2006). Another work is (Qazi et al., 2007) in which the authors use a directed acyclic graph for the class subgraph, an empty graph for the features, and a bridge subgraph where features receive arcs from some class variables, without sharing any of them. (Bielza et al., 2011) present the most general models since any Bayesian network structure is allowed in the three subgraphs. Moreover they use all the possibilities for learning from data: wrapper, filter and hybrid score+search strategies.

Finally, another algorithm for multi-label classification is named RAKEL (Random K-Labelsets) (Tsoumakas and Katakis, 2007). RAKEL obtains  $m$  random subsets of size  $k$  and for each one a power set is constructed. A voting scheme with a user defined threshold determines the classification. One disadvantage of this approach is the random process used for obtaining the subsets, since no information is guiding the process.

### 3.2 Chain Classifiers

(Read et al., 2011) introduce chain classifiers as an alternative method for multi-label classification that incorporates class dependencies, while it tries to keep the computational efficiency of the binary relevance approach. Chain classifiers consist of  $d$  binary classifiers which are linked in a chain, such that each classifier incorporates the classes predicted by the previous classifiers as additional attributes. Thus, the feature vector for each binary classifier,  $L_i$ , is extended with the labels (0/1) of all previous classifiers in the chain. Each classifier in the chain is trained to learn the association of label  $l_i$  given the features augmented with all previous binary predictions in the chain,  $l_1, l_2, \dots, l_{i-1}$ . For classification, it starts at  $L_1$ , and propagates along the chain such that for  $i \in \mathcal{L}$  (where  $\mathcal{L} = \{l_1, l_2, \dots, l_d\}$ ) it predicts  $p(l_i | \mathbf{x}, l_1, l_2, \dots, l_{i-1})$ . As in the binary relevance approach, the class vector is determined by combining the outputs of all the binary classifiers in the chain. They combine several chain classifiers by changing the order for the labels, building an ensemble of chain classifiers. The

final label vector is obtained using a voting scheme; each label  $l_i$  receives a number of votes from the  $m$  chain classifiers, and a threshold is used to determine the final predicted multi-label set. They used support vector machines as the base binary classifier.

### 3.3 Bayesian Chain Classifiers

Given a multi-label classification problem with  $d$  classes, a Bayesian chain classifier (BCC)(Zaragoza et al., 2011) uses  $d$  classifiers, one per class, linked in a chain. The objective of this problem can be posed as finding a joint distribution of the classes  $\mathbf{C} = (C_1, C_2, \dots, C_d)$  given the attributes  $\mathbf{x} = (x_1, x_2, \dots, x_l)$ :  $p(\mathbf{C}|\mathbf{x}) = \prod_{i=1}^d p(C_i|\mathbf{pa}(C_i), \mathbf{x})$  where  $\mathbf{pa}(C_i)$  represents the parents of class  $C_i$ . In this setting, a chain classifier can be constructed by inducing first the classifiers that do not depend on any other class and then proceed with their sons, according to the dependence structure which can be represented as a Bayesian network.

(Zaragoza et al., 2011) build a tree-structured dependency model. They simplify the problem by considering the marginal dependencies between classes, using (Chow and Liu, 1968) algorithm, that is, a *maximum weight undirected spanning tree (MWST)*. Then they take a class (node) as root of a tree and assign directions to the arcs starting from this root node to build a directed tree. The chaining order of the classifiers is given by traversing the tree following an ancestral ordering. Based on this order they construct chain classifiers in which each base classifier incorporates one additional attribute, its parent class in the tree.

### 3.4 Other approaches

In (Kang et al., 2006) the authors describe a method to explicitly model correlations between class labels. The multi-label problem is posed as an optimization problem that considers how similar the training samples are with the testing samples considering subsets of classes at the same time. It needs to assign a weight to the classes given by their frequency. The method follows a greedy strategy that depends

on the relative order of the weights of the classes which are given in reverse order of the class frequency. In (Ji et al., 2010) the authors describe a method to extract shared structures (subspaces) in multi-label classification problems to capture the correlation information among classes. In this framework, a subspace is assumed to be shared among multiple labels, and a linear transformation is computed to discover this subspace. The shared structure is obtained by solving an eigenvalue problem using a regularization term to reduce the complexity.

In (Zhang and Zhang, 2010) the authors use a Bayesian network to encode the conditional dependencies of the labels as well as the feature set, with the feature set as the common parent of all labels. Their approach first constructs classifiers for all the labels independently. This produces some errors. They then learn a Bayesian network structure guided by these errors. Finally, they construct a new classifier for each label incorporating their parents in the Bayesian network as additional features.

## 4 Hybrid Binary–Chain Classifiers

Constructing chain classifiers with a random class order, as in (Read et al., 2011), can introduce irrelevant information and unnecessary computation, as many of the classes used as additional attributes in the chain may be independent of the current class. On the other hand, trying to find the probabilistic dependencies among classes given a set of attributes, can be computationally expensive. In this paper we find the independent groups of classes, using a simple correlation analysis, and use that information to significantly reduce the computation time while achieving competitive predictive performance results.

### 4.1 Correlation and Class Dependency

Correlation and statistical independence are two basic concepts denoting a relation among events. While both concepts express a notion of the link among processes they are fundamentally different (Mari and Kotz, 2001). It is a common mistake to deduce statistical independency from a lack of correlation and even worse,

to infer statistical dependency from a strong correlation. The latter is not true and the former is valid as long as the relationship is linear, although zero correlation can be obtained when non-linear relations exist. In this paper, we assume a linear relationship among classes to identify independencies. Of course, other measures can be used instead such as mutual information for instance.

### 4.2 HBCC

To build a Hybrid Binary–Chain Classifier (HBCC) we start by obtaining the pairwise linear correlation coefficients between each pair of classes. Uncorrelated classes are considered independent and consequently, independent classifiers can be built from them. Correlated classes are grouped together and for each group a chain classifier is built. Groups with common members are merged into a single group. In the worst case, all the classes are grouped together and therefore this would degenerate in a chain using all the classes as proposed in (Read et al., 2011). By grouping only the correlated classes, we substantially reduce the computation time and eliminate irrelevant information from independent classes that is normally included in chain classifiers. Also, by focusing on identifying independent classes, we can apply a simpler approach based on correlation, rather than a more sophisticated approach such as structure learning of Bayesian networks. Algorithm 1 provides a description of the HBCC algorithm. The algorithm requires a parameter,  $\lambda$ , which defines the threshold used to consider two variables (classes) independent. If this threshold is too high the algorithm reduces to the binary relevance approach, if it is too low the result is a chain classifier (Read et al., 2011).

## 5 Experiments and Results

The experiments were performed on 9 different benchmark multi-label datasets<sup>1</sup>; each of them with different dimensions ranging from 6 to 983 labels, and from about 600 examples to more

<sup>1</sup>The data sets can be found at [mulan.sourceforge.net/datasets.html](http://mulan.sourceforge.net/datasets.html) and at [www.cs.waikato.ac.nz/~jmr30/\#datasets](http://www.cs.waikato.ac.nz/~jmr30/\#datasets).

---

**Algorithm 1** Learning a Hybrid Binary-Chain Classifier.

---

**Input:** a multi-label dataset ( $BD$ ), a threshold value ( $\lambda$ )  
 Obtain a correlation matrix ( $CM$ ) among all classes in  $BD$

**for** each  $e \in CM$  **do**

- if**  $e < \lambda$  **then**

  - $e \leftarrow 0$

- else**

  - $e \leftarrow 1$

- end if**

**end for**

Obtain  $G$  groups of correlated classes

**for** each  $g \in G$  **do**

- if**  $\text{size}(g) = 1$  **then**

  - Create a Binary classifier for class  $g$

- else**

  - Create a Chain Classifier using the classes in  $g$ .

- end if**

**end for**

---

than 40,000. All class variables of the datasets are binary, however, in some of the datasets the feature variables are numeric. In these cases we used a static, global, supervised and top-down discretization algorithm (Cheng-Jung et al., 2008). The details of the datasets are summarized in Table 1.

For the purpose of comparison we used four different multi-label precision measures (Bielza et al., 2011; Read et al., 2011):

1. *Mean accuracy* over the  $d$  class variables (accuracy per label):

$$\text{M-Acc} = \frac{1}{d} \sum_{j=1}^d \text{Acc}_j = \frac{1}{d} \sum_{j=1}^d \frac{1}{N} \sum_{i=1}^N \delta(c'_{ij}, c_{ij}) \quad (1)$$

where  $\delta(c'_{ij}, c_{ij}) = 1$  if  $c'_{ij} = c_{ij}$  and 0 otherwise. Note that  $c'_{ij}$  denotes the  $C_j$  class value outputted by the model for case  $i$  and  $c_{ij}$  is its true value.

2. *Global accuracy* over the  $d$ -dimensional

Table 1: Multi-Label datasets used in the experiments.  $N$  is the size of the dataset,  $d$  is the number of binary classes or labels,  $m$  is the number of features. \* indicates numeric attributes.

| No. | Dataset   | $N$   | $d$ | $m$  | Type    |
|-----|-----------|-------|-----|------|---------|
| 1   | Emotions  | 593   | 6   | 72*  | Music   |
| 2   | Scene     | 2407  | 6   | 294* | Vision  |
| 3   | Yeast     | 2417  | 14  | 103* | Biology |
| 4   | Medical   | 978   | 45  | 1449 | Text    |
| 5   | Enron     | 1702  | 53  | 1001 | Text    |
| 6   | TMC2007   | 28596 | 22  | 500  | Text    |
| 7   | Bibtex    | 7395  | 159 | 1836 | Text    |
| 8   | MediaMill | 43907 | 101 | 120* | Media   |
| 9   | Delicious | 16105 | 983 | 500  | Text    |

class variable (accuracy per example):

$$\text{G-Acc} = \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{c}'_i, \mathbf{c}_i) \quad (2)$$

where  $\delta(\mathbf{c}'_i, \mathbf{c}_i) = 1$  if  $\mathbf{c}'_i = \mathbf{c}_i$  and 0 otherwise. Therefore, we call for a total coincidence on all of the components of the vector of predicted classes and the vector of real classes.

3. *Multilabel accuracy* as defined in (Tsoumakas and Katakis, 2007):

$$\text{ML-Acc} = \frac{1}{N} \sum_{i=1}^N \frac{\mathbf{c}_i \wedge \mathbf{c}'_i}{\mathbf{c}_i \vee \mathbf{c}'_i} \quad (3)$$

In this measure, accuracy is micro-averaged across all examples.

4. *F measure* is the harmonic mean between precision and recall.

$$\text{F-measure} = \frac{1}{d} \sum_{i=1}^d \frac{2 \times p_j \times r_j}{(p_j + r_j)} \quad (4)$$

where  $p_j$  and  $r_j$  are the precision and recall for all  $C_j$ . Here, the accuracy is calculated per label and then averaged.

We estimated the performance measures using 10-fold cross-validation<sup>2</sup>. We used the Naive

<sup>2</sup>In a Core 2 Duo at 2.4 GHz with 8 GB of RAM.

Bayes, J48 and SVM implementations of Weka (Witten et al., 2011) software.

We compared HBCC against *binary relevance* (all the classes are considered to be independent of each other) and chain classifiers (as proposed in (Read et al., 2011)). We applied the Kruskal-Wallis test for testing statistical significance, which is an extension of the Wilcoxon rank sum test for more than two groups ( $\alpha = 5\%$ ).

We first performed experiments with different values for  $\lambda$  (threshold value for building the correlation matrix) ranging from  $\lambda = 0.1$  with almost all classes grouped together ( $\approx$  Chain-NB) to  $\lambda = 0.9$  with almost all classes independent ( $\approx$  BR). We only show the average results for all the datasets and for each performance metric (see Table 2).

As can be seen from the experiments, considering some form of correlation improves the performance over the binary relevance method and is very competitive with chain classifiers.

For the next experiments we set  $\lambda = 0.6$ , which is the value that obtained the best results on average, for all the datasets.

Table 3 summarizes the accuracy results for HBCC, Chain-NB classifiers (Read et al., 2011), and binary relevance (BR) methods. Since dataset Delicious did not produce an output using Chain-NB after 48 hrs, the results of this dataset are not used for obtaining the average of the three approaches. From the results we can observe that HBCC obtained the best score in Mean and Global measures. Also there are some results in which HBCC obtained statistical significant difference with respect to the other two approaches.

We also quantified how much saving, in terms of attributes and in terms of processing time, are obtained with HBCC in comparison with a chain classifier.

Table 4 shows the number of extra attributes added in all the classifiers constructed for each dataset and the average size of the groups obtained with HBCC. From this table we can see that our proposed method greatly reduces the number of added attributes when compared to Chain-NB approach.

Finally, Table 5 indicates the processing time

Table 4: Number of attributes added when classifying a dataset using Correlated-Based chain classifier (HBCC) and Chain-NB approach. Also the average size of the chains obtained by HBCC are presented.

| Data set  | <i>Attributes added</i> |          | Average chain size<br>HBCC |
|-----------|-------------------------|----------|----------------------------|
|           | HBCC                    | Chain-NB |                            |
| Emotions  | 1                       | 15       | 1.16                       |
| Scene     | 0                       | 15       | 1                          |
| Yeast     | 12                      | 91       | 1.85                       |
| Medical   | 0                       | 990      | 1                          |
| Enron     | 12                      | 1378     | 1.22                       |
| TMC2007   | 1                       | 231      | 1.04                       |
| Bibtex    | 26                      | 12561    | 1.16                       |
| MediaMill | 59                      | 5050     | 2.71                       |
| Delicious | 4568                    | 482653   | 5.64                       |
| Average   | 519.88                  | 55887.11 | 1.86                       |

of a HBCC, including the time to evaluate the correlation matrix using Matlab; which is compared to the time obtained by the Chain-NB and binary relevance approaches. The savings in time obtained by the proposed algorithm are notable and becomes more pronounced with larger datasets. In particular, for the largest dataset, our approach is at least 5 times faster in comparison to Read's approach.

Table 6 shows the performance of the proposed approach with two other base classifiers, C4.5 and SVM (implementations taken from Weka). We only show the average results for all the datasets and for each of the performance metrics. With C4.5 classifier our approach obtained the best results for all the measures. With SVM classifier HBCC obtained the best results in two measures and competitive results in the rest.

## 6 Conclusions and Future Work

In this paper we have introduced a Hybrid Binary-Chain Classifier for multi-label classification. The proposed approach is simple and easy to implement, and yet is highly competitive in terms of classification performance against a chain classifiers, and more efficient. We consider that this approach provides a practical and powerful alternative for building multidimensional classifiers.

Table 2: Average results with different threshold values for all the datasets and each performance metric.

|           | 0.1   | 0.2          | 0.3   | 0.4   | 0.5          | 0.6          | 0.7   | 0.8   | 0.9   | BR    | Chain-NB     |
|-----------|-------|--------------|-------|-------|--------------|--------------|-------|-------|-------|-------|--------------|
| Mean      | 0.859 | 0.859        | 0.860 | 0.859 | 0.860        | <b>0.860</b> | 0.860 | 0.859 | 0.858 | 0.858 | 0.859        |
| Global    | 0.210 | <b>0.210</b> | 0.209 | 0.207 | 0.208        | 0.208        | 0.208 | 0.207 | 0.207 | 0.207 | 0.200        |
| ML        | 0.411 | 0.412        | 0.412 | 0.410 | 0.411        | <b>0.412</b> | 0.411 | 0.411 | 0.410 | 0.410 | <b>0.413</b> |
| F-measure | 0.388 | 0.388        | 0.389 | 0.389 | <b>0.390</b> | 0.389        | 0.390 | 0.388 | 0.387 | 0.387 | <b>0.392</b> |

Table 3: Classification accuracies for HBCC, (Read et al., 2011) approach (using Naive Bayes as base classifier) and binary relevance (BR) method. A “†” indicates statistically significant differences between HBCC and Chain-NB and a “\*” indicates statistically significant differences between HBCC and BR.

| Data set  | Mean Accuracy        |               |                | Global Accuracy |               |                |
|-----------|----------------------|---------------|----------------|-----------------|---------------|----------------|
|           | HBCC                 | BR            | Chain-NB       | HBCC            | BR            | Chain-NB       |
| Emotions  | 0.8454               | <b>0.8460</b> | 0.8449         | 0.3879          | <b>0.3895</b> | 0.3879         |
| Scene     | <b>0.9058</b>        | <b>0.9058</b> | 0.9055         | <b>0.5343</b>   | <b>0.5343</b> | 0.5301         |
| Yeast     | <b>0.8727*</b>       | 0.8641        | 0.8673         | <b>0.2768†</b>  | 0.2702        | 0.2586         |
| Medical   | <b>0.9746</b>        | <b>0.9746</b> | 0.9739         | <b>0.2648</b>   | <b>0.2648</b> | 0.2587         |
| Enron     | <b>0.7811</b>        | <b>0.7811</b> | 0.7762         | <b>0.0012†</b>  | 0.0006        | 0.0000         |
| TMC2007   | <b>0.8888</b>        | <b>0.8888</b> | 0.8803         | <b>0.1435†</b>  | <b>0.1435</b> | 0.1136         |
| Bibtex    | <b>0.9130</b>        | 0.9126        | 0.9107         | <b>0.0594</b>   | 0.0592        | 0.0549         |
| MediaMill | 0.7003               | 0.6963        | <b>0.7168†</b> | <b>0.0003</b>   | <b>0.0003</b> | 0.0001         |
| Delicious | <b>0.8937</b>        | 0.8871        | -              | 0.0000          | 0.0000        | -              |
| Average   | <b>0.8602</b>        | 0.8587        | 0.8594         | <b>0.2085</b>   | 0.2078        | 0.2005         |
| Data set  | Multi-Label Accuracy |               |                | F-measure       |               |                |
|           | HBCC                 | BR            | Chain-NB       | HBCC            | BR            | Chain-NB       |
| Emotions  | 0.6689               | <b>0.6695</b> | 0.6679         | 0.7655          | <b>0.7663</b> | 0.7646         |
| Scene     | <b>0.7161</b>        | <b>0.7161</b> | 0.7157         | 0.7845          | 0.7845        | <b>0.7848</b>  |
| Yeast     | <b>0.6733*</b>       | 0.6571        | 0.6699         | 0.5935          | 0.5730        | <b>0.6307†</b> |
| Medical   | <b>0.3663</b>        | <b>0.3663</b> | 0.3596         | <b>0.0865</b>   | 0.0865        | 0.0840         |
| Enron     | <b>0.1937</b>        | 0.1935        | 0.1869         | <b>0.1460</b>   | 0.1459        | <b>0.1472</b>  |
| TMC2007   | <b>0.4829</b>        | <b>0.4829</b> | 0.4333         | <b>0.4717</b>   | 0.4717        | 0.4522         |
| Bibtex    | <b>0.1879†</b>       | 0.1872        | 0.1755         | <b>0.1851</b>   | 0.1840        | 0.1817         |
| MediaMill | 0.0114               | 0.0110        | <b>0.0982†</b> | 0.0868          | 0.0847        | <b>0.0941</b>  |
| Delicious | <b>0.1352</b>        | 0.1249        | -              | <b>0.0702</b>   | 0.0620        | -              |
| Average   | 0.4126               | 0.4105        | <b>0.4134</b>  | 0.3899          | 0.3871        | <b>0.3924</b>  |

Table 5: Time (seconds) required to build a multi-label classifier using HBCC, Chain-NB and BR approaches. The first column shows the time to build a correlation matrix using Matlab, the third column shows the total time used for HBCC, the fourth shows the total time used for the Chain-NB approach and the fifth shows the total time for the BR approach.

| Data set  | HBCC   |          | Chain-NB |            | BR       |
|-----------|--------|----------|----------|------------|----------|
|           | Matrix | HBCC     | Total    | Total      |          |
| Emotions  | 0.01   | 0.27     | 0.28     | 0.44       | 0.27     |
| Scene     | 0.01   | 1.34     | 1.35     | 2.46       | 1.30     |
| Yeast     | 0.04   | 1.57     | 1.61     | 5.55       | 1.29     |
| Medical   | 0.38   | 23.10    | 23.48    | 91.30      | 22.28    |
| Enron     | 0.56   | 41.08    | 41.63    | 245.93     | 41.14    |
| TMC2007   | 0.28   | 122.00   | 122.28   | 228.65     | 120.24   |
| Bibtex    | 7.16   | 1064.58  | 1071.74  | 4770.54    | 1002.00  |
| MediaMill | 10.09  | 714.61   | 724.71   | 12120.53   | 615.646  |
| Delicious | 364.30 | 30638.98 | 31003.28 | >172800.00 | 25200.02 |
| Total     | 382.83 | 32607.52 | 32990.36 | >190265.40 | 27004.19 |

Table 6: Performance with C4.5 and SVM

|             | C4.5          |        |        | SVM           |        |               |
|-------------|---------------|--------|--------|---------------|--------|---------------|
|             | HBCC-0.6      | BR     | Chain  | HBCC-0.6      | BR     | Chain         |
| Mean Acc.   | <b>0.9108</b> | 0.9023 | 0.9087 | 0.9280        | 0.9230 | <b>0.9283</b> |
| Global Acc. | <b>0.2239</b> | 0.2178 | 0.2225 | 0.3509        | 0.3451 | <b>0.3556</b> |
| ML Acc.     | <b>0.4072</b> | 0.3900 | 0.4031 | <b>0.5760</b> | 0.5658 | 0.5718        |
| F-measure   | <b>0.3618</b> | 0.3246 | 0.3405 | <b>0.5173</b> | 0.5041 | 0.5125        |

As future work we plan to incorporate Bayesian chain classifiers for each group. Also we plan to use other clustering alternatives like spectral clustering or agglomerative clustering, as well as other measures such as Hilbert Schmidt independence criterion.

## References

- C. Bielza, G. Li, and P. Larrañaga. 2011. Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*.
- T. Cheng-Jung, L. Chien-I, and Y. Wei-Pang. 2008. A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences*, (178):714–731.
- C. Chow and C. Liu. 1968. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467.
- S. Ji, L. Tang, S. Yu, and J. Ye. 2010. A shared-subspace learning framework for multi-label classification. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(2):8.
- F. Kang, R. Jin, and R. Sukthankar. 2006. Correlated label propagation with application to multi-label learning. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1719–1726. IEEE.
- D.D. Mari and S. Kotz. 2001. *Correlation and dependence*. Imperial College Press.
- M. Qazi, G. Fung, S. Krishnan, R. Rosales, H. Steck, R. Bharat Rao, D. Poldermans, and D. Chandrasekaran. 2007. Automated heart wall motion abnormality detection from ultrasound images using bayesian networks. *International Joint Conference on Artificial Intelligence*, pages 519–525.
- J. Read, B. Pfahringer, G. Holmes, and E. Frank. 2011. Classifier chains for multi-label classification. *Machine Learning*, 85:333–359. 10.1007/s10994-011-5256-5.
- S. E. Shimony. 1994. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410.
- Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13.
- Linda C. van der Gaag and Peter R. de Waal. 2006. Multi-dimensional bayesian network classifiers. In *Third European Conference on Probabilistic Graphic Models*, pages 107–114.
- C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214.
- I.H. Witten, E. Frank, and M.A. Hall. 2011. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.
- J.H. Zaragoza, L.E. Sucar, EF Morales, P. Larrañaga, and C. Bielza. 2011. Bayesian chain classifiers for multidimensional classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*.
- M.L. Zhang and K. Zhang. 2010. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 999–1008. ACM.
- M. Ling Zhang and Z. Hua Zhou. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

# Estimation of effect size posterior using model averaging over Bayesian network structures and parameters

Gabor Hullam

Budapest University of Technology and Economics, Hungary

gabor.hullam@mit.bme.hu

Peter Antal

Budapest University of Technology and Economics, Hungary

peter.antal@mit.bme.hu

## Abstract

Bayesian networks provide multiple advantages in feature subset analysis, but the parametric properties of Bayesian networks are typically neglected in inductive approaches, as they are focused on existential inference of relevant variables. However, in many application fields, particularly in biomedicine, certain parametric properties, such as effect size measures, are essential and widely used in interpretation, because they provide a more detailed characterization of relevance. To cope with multiple hypothesis testing, the Bayesian statistical framework became popular, but the posterior distribution of these effect size measures derived by Bayesian model averaging over structures are analytically not tractable, thus their estimation requires Monte Carlo simulation methods. In this paper we compare the structural (existential) and parametric (quantitative) concepts of Bayesian relevance and overview Bayesian approaches to the estimation of effect size posteriors. Furthermore, we compare the high probability density regions of these posteriors with traditional frequentist confidence intervals. Methods are illustrated on an artificial data set simulating a genetic association study.

## 1 Introduction

The application of graphical models, especially Bayesian networks, became increasingly popular in the field of biomedicine and genomics, especially in feature subset analysis when the need for modeling potentially complex dependency structures between genomic, environmental, and clinical factors and disease state indicators emerged. Recently, it became apparent that the simple phenotype approach, which is applied in most genome-wide association studies, has considerable limits. Analyses based on standard statistical methods using the hypothesis testing framework were frequently unsuccessful. This phenomenon induced an extensive research for applicable methods, capable of handling the needs of genetic research.

Bayesian network based learning methods ap-

peared in this field with the promise of revealing the possibly complex dependency relationships between the investigated factors. Several Bayesian network based methods focus on the learning of some structural properties or the whole structure from data. Even though learning structural properties provides invaluable information on the relationships between factors, and enables the selection of relevant factors with respect to a specified target, it neglects parametric properties such as effect size. One might argue, that one of the main benefits of Bayesian networks with conjugate priors in the context of structure learning is that the parametric level can be bypassed, i.e. analytically averaged out. However, in several domains, such as biomedicine, effect size as a quantitative descriptor plays a central role in the interpretation of the results from feature subset analysis.

Therefore, a Bayesian effect size measure applicable within the same framework as the structure learning methods could enhance the interpretation of such results. Earlier approaches include a Bayesian estimation of log odds ratios in presence of prior information (Demirhan and Hamurkaroglu, 2008), a Bayesian method for the non-informative prior case (Rahardja et al., 2010), and an inference of posteriors of mutual information using a Bayesian framework (Hutter and Zaffalon, 2005).

Although there are many possible measures for the relevance of a variable, three main approaches can be distinguished: the association based approach neglecting structural aspects, the causal approach assuming a fixed structure, and the existential approach averaging over structures. The widespread association based approach uses effect size measures that do not take structural, multivariate relationships into account. Odds ratio (OR) is the most widely used such measure, especially in case-control studies.

The causal approach measures the effect of a variable  $X$  on another  $Y$  given a structure describing causal relationships, i.e. the nature and the strength of a relationship between variables  $X$  and  $Y$ . Structural equation modeling and the average causal effect measure (Pearl, 2000) are two related methods providing a measure of effect size assuming a known causal structure. The obvious drawback of these methods is the lack of learning structures.

The structural (existential) uncertainty based approach uses Bayesian networks, which provide a graph based language for encoding relevance and representing dependency relationships. Assuming a Bayesian framework, a high posterior for strong relevance indicates a non-mediated relationship between  $X$  and  $Y$ , i.e. it indicates a close structural connection (Friedman and Koller, 2003).

Note that parametric relevance does not imply structural relevance (e.g. strong relevance), and vice versa a relatively high posterior of strong relevance does not imply relatively high parametric relevance, e.g. a quantitative measure of effect size e.g. odds ratio, denoted as

$\text{OR}(X, Y)$ , may indicate the relevance of  $X$  with respect to a selected target  $Y$  just by being over a certain threshold e.g.  $\text{OR}(X, Y) \geq 2.5$  (the structural relation between  $X$  and  $Y$  is unknown and has no influence on this aspect of relevance). This shows that the structural strong relevance and parametric effect size aspects appear to be two separate dimensions of relevance. The main goal of this paper is to connect these two aspects of relevance by introducing a structure based Bayesian effect size measure  $OR(X, Y|\theta, G)$  based on  $p(\theta, G|D_N)$ , where  $\theta$  denotes the parametrization and  $G$  the structure of an underlying BN, and  $D_N$  denotes data (i.e.  $OR(X, Y|\theta, G)$  is a random variable with distribution  $p(\theta, G|D_N)$ ).

A further reason behind the structure based Bayesian effect size measure is that there is no closed form for the posterior of effect size measures, such as odds ratio. Thus, the posterior has to be estimated, e.g. by sampling the log-odds ratio of a Dirichlet distribution based on data. Another possible solution is to use the structural properties of BNs to guide the estimation process. Though instead of learning a whole Bayesian network from the data, the learning of relevant variables with respect to a target (i.e. the Markov blanket of the target) is sufficient.

In the following section we discuss the importance of structural properties of BNs and their relation to relevance and to effect size. Then we introduce the Bayesian odds ratio and describe various methods for its estimation. Finally, we demonstrate the properties of Bayesian odds ratio using an artificial data set.

## 2 Strong relevance and its relation to structural properties of BNs

Relevance is a central concept in all processes in which learning is involved. In case of learning structural properties of a Bayesian network from data, the identification of relevant elements is essential. In realistic cases, with at least a hundred variables, the learning of the whole network is practically untractable. A possible approach is to learn smaller substruc-

tures or structural properties that are statistically better confirmed.

Several methods were created for this purpose, starting with a Bayesian inference over structural properties of Bayesian networks (Buntine, 1991; Cooper and Herskovits, 1992). In (Madigan et al., 1996) a Markov Chain Monte Carlo (MCMC) scheme was proposed to approximate such Bayesian inference. Subsequently, Friedman et al. reported an MCMC scheme over the space of orderings (Friedman and Koller, 2003). Then in (Koivisto and Sood, 2004) a method to perform exact full Bayesian inference over modular features was introduced, and an ad hoc randomized approach was reported in (Pena et al., 2007). Recent applications in genetic association studies were discussed in (Jiang et al., 2010; Xing et al., 2011).

Despite the leading role of relevance, its relation to structural properties and to the concepts of association and effect size, seemed unclarified in some cases. Thus we focused our efforts towards investigating various structural properties of BNs related to relevance. We demonstrated the Bayesian application of Bayesian networks in relevance analysis (Antal et al., 2006), and proposed a Bayesian network based Bayesian multilevel analysis of relevance (Antal et al., 2008a). We carried out a comparative study of BN-BMLA against other methods in (Hullám et al., 2010), and applied the BN-BMLA method in a candidate gene association study of asthma (Ungvari et al., 2012).

Here we provide a brief summary of the most important aspects of relevance. First of all, the concept of relevance can be defined in multiple ways. On the one hand, it can be defined specific to the applied model class used as a predictor, the optimization algorithm, the data set, and the loss function (Kohavi and John, 1997).

**Definition 1** (Strong and weak relevance). A feature  $X_i$  is strongly relevant to  $Y$ , if there exists some  $X_i = x_i, Y = y$  and  $s_i = x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$  for which  $p(x_i, s_i) > 0$  such that  $p(y|x_i, s_i) \neq p(y|s_i)$ . A feature  $X_i$  is weakly relevant, if it is not strongly relevant, and there exists a subset of features  $S'_i$  of  $S_i$  for

which there exists some  $x_i, y$  and  $s'_i$  for which  $p(x_i, s'_i) > 0$  such that  $p(y|x_i, s'_i) \neq p(y|s'_i)$ . A feature is relevant, if it is either weakly or strongly relevant; otherwise it is irrelevant.

In contrast, association as a relevance type only means that there is a relationship between  $X_i$  and  $Y$  such that they are statistically dependent  $p(y|x_i) \neq p(y)$ , however association does not provide information to discriminate between strong and weak relevance. This is also indicated by the possibility that  $X_i$  is strongly relevant, yet there is no association found with  $Y$ . In other words, strong relevance entails strong criteria on the structure of an underlying BN, whereas association does not necessarily. Finally, another difference between strong relevance and association is that the strength of association can be characterized by several measures, such as odds ratio or mutual information, whereas there are no such quantitative descriptors for the strength of strong relevance. These differences also confirm the necessity of dual characterization of relevance. For this purpose we propose the structure based Bayesian odds ratio, which is a combination of effect size and structural information.

The relationship between structural properties of BNs and strong relevance is revealed by the following model-based, probabilistic definition of Markov blankets (Pearl, 1988) and by a subsequent theorem.

**Definition 2** (Markov blanket). A set of variables  $\mathbf{X}' \subseteq \mathbf{V}$  is called a *Markov blanket* set of  $X_i$  w.r.t. the distribution  $p(\mathbf{V})$ , if  $(X_i \perp\!\!\!\perp \mathbf{V} \setminus \mathbf{X}' | \mathbf{X}')$ , where  $\perp\!\!\!\perp$  denotes conditional independence.

The following theorem gives a sufficient condition for the unambiguous BN representation of the relevant structural properties (Tsamardinos and Aliferis, 2003).

**Theorem 1.** *For a distribution  $p$  defined by Bayesian network  $(G, \theta)$  the variables  $\text{bd}(Y, G)$  form a Markov blanket of  $Y$ , where  $\text{bd}(Y, G)$  denotes the set of parents, children and the children's other parents for  $Y$  (Pearl, 1988). If the distribution  $p$  is stable with respect to the DAG  $G$ , then  $\text{bd}(Y, G)$  forms a unique and mini-*

mal Markov blanket of  $Y$ , denoted as  $\text{MBS}_p(Y)$ . Furthermore,  $X_i \in \text{MBS}_p(Y)$  iff  $X_i$  is strongly relevant.

We also refer to  $\text{bd}(Y, G)$  as the Markov blanket set for  $Y$  in  $G$  using the notation  $\text{MBS}(Y, G)$  by the implicit assumption that  $p$  is Markov compatible with  $G$ . Based on  $\text{bd}(Y, G)$  a pairwise relation can be defined indicating whether  $X_j$  is a member of  $\text{bd}(Y, G)$ . This relation is called Markov blanket membership and it is denoted as  $\text{MBM}(X_j, Y)$ .

In other words, this means that an  $\text{MBS}(Y, G)$  contains all the strongly relevant variables  $X_j$  with respect to  $Y$ . Furthermore, estimation of effect size measure for the strongly relevant variables in  $\text{MBS}(Y, G)$  depends only from parameters in the Markov Blanket sub-Graph under multinomial sampling and global parameter independence, thus we can rely on this set of variables and parameters instead of taking the whole BN structure into consideration (see Eq. 4).

### 3 Bayesian effect size

In several fields, such as biomedicine and genetics, the most frequently used effect size measure is the odds ratio (Balding, 2006).

Let  $X_1, X_2, \dots, X_n$  denote discrete variables that encode single nucleotide polymorphism (SNP) states 0,1,2 that refer to common (wild) homozygote, heterozygote, rare (mutant) homozygote genotypes respectively. Then  $X_i^{(s)}$  denotes SNP  $X_i$  in state  $s$ . In case of a disease indicator  $Y$ , i.e.  $Y^{(0)}$ : non-affected (control),  $Y^{(1)}$ : affected (case), an *odds* is defined as

$$\text{Odds}(X_i^{(s)}) = \frac{p(Y^{(1)}|X_i^{(s)})}{p(Y^{(0)}|X_i^{(s)})} \quad (1)$$

Consequently an *odds-ratio* e.g. heterozygous (1) versus common homozygous (0) is given as

$$\text{OR}(X_i^{(1,0)}) = \frac{\text{Odds}_{X_i^{(1)}}}{\text{Odds}_{X_i^{(0)}}} \quad (2)$$

Although odds ratio is a simple measure, the computation of significance of its value (and its

confidence interval) is plagued by multiple testing. A possible approach is to apply univariate Bayesian methods using various priors as suggested by (Stephens and Balding, 2009). However, this was not extended to the multivariate case, i.e. all SNPs were treated as independent entities, thus prohibiting the analysis of joint effects.

A possible multivariate Bayesian approach is to utilize the underlying BN  $(G, \theta)$  for odds ratio computation  $\text{OR}(\theta)$ . However, assuming a Dirichlet prior  $\theta \sim \text{Dir}(X_i, \alpha_i)$ , this computation is analytically non-tractable. Therefore, we propose to sample the parameters in the 'relevant part' of the BN, instead of the whole structure. The structural property in which the relevant structural elements (with respect to a target) are encoded is called the Markov blanket graph (Acid et al., 2005; Antal et al., 2006).

**Definition 3** (Markov blanket graph). A Markov blanket graph  $\text{MBG}(Y, G)$  of variable  $Y$  is a subgraph of a Bayesian network structure  $G$ , which contains the nodes of the Markov blanket set of  $Y$ , that is  $\text{MBS}(Y, G)$  and the incoming edges into  $Y$  and its children. Given a target node, which corresponds to the target variable  $Y$ ,  $\text{MBG}(Y, G)$  as a (sub)graph structure consists of nodes that are (1) parents of  $Y$ , (2) children of  $Y$  or (3) "other parents" of the children of  $Y$ .

Note, that from all the possible edges between these nodes,  $\text{MBG}(Y, G)$  only contains those that end in  $Y$  or one of its children. Mechanism boundary graph is another term for Markov blanket graph due to its significant role in describing causal relationships (i.e. mechanisms). As we discussed in section 1, the causal or mechanism based approach is one of the main approaches to effect size computation. According to which, given a causal relationship  $A \rightarrow B$  the effect size measure defines the amount of effect a value of variable  $A$  has on specific values of  $B$ . Since the Bayesian network representation can be interpreted in a causal context under the Causal Markov Condition (for details see (Pearl, 2000)) then an edge between vertices  $X_i$  and  $X_j$  can denote a causal relationship  $X_i \rightarrow X_j$ .

From this perspective  $MBG(Y, G)$  describes the direct causal relationships of  $Y$  (i.e. edges from nodes of type (1) and edges to nodes of type (2)), and also some of the indirect relationships of  $Y$  that form a special dependency pattern, called a *v-structure* (Pearl, 2000).

Note that  $MBG(Y, G)$  includes all the mechanisms, in which  $Y$  is involved. This property makes the Markov blanket graph an ideal candidate to serve as a base for measuring effect size both from the causal aspect and the structural relevance aspect. The latter is due to the fact that the nodes of  $MBG(Y, G)$  represent the variables of  $MBS(Y, G)$ , that is the strongly relevant variables with respect to  $Y$ .

Since the goal is to measure the effect of a factor  $X_i$  (e.g. SNPs) on a target variable  $Y$  (e.g. disease susceptibility) the fact that  $X_i$  is a member of  $MBG(Y, G)$  is a relevant information. If  $X_i$  is a member, then the probability of a certain value of the target  $Y$  (e.g. in case of  $Y$  as a disease status the values are: "case" and "control") can be estimated based on the  $MBG(Y, G)$  and a specific instantiation of  $X_i$ . This in turn allows the computation of a Markov blanket graph based odds ratio

$$\text{OR}(X_i^{(q,r)} | X_i \in MBG(Y, G)) = \frac{p(Y^{(1)}|mbg, X_i^{(q)})}{p(Y^{(0)}|mbg, X_i^{(q)})} \cdot \frac{p(Y^{(0)}|mbg, X_i^{(r)})}{p(Y^{(1)}|mbg, X_i^{(r)})} \quad (3)$$

where  $mbg$  refers to  $X_{j \neq i} \in MBG(Y, G)$ . In the opposite case, when  $X_i$  is not a member of  $MBG(Y, G)$  then its effect on  $Y$  is negligible, because the mechanisms in  $MBG(Y, G)$  shield  $Y$  from the effect of  $X_i$ .

#### 4 The estimation of Bayesian odds ratio

In terms of BN learning, the first step of effect size estimation is the learning of structures, that is identifying the direct and indirect relationships between entities of a domain, i.e. between the variables of a data set. In order to characterize the relationships, a second step, the learning of conditional distribution parameters

is needed. As we discussed in section 2, instead of learning the whole BN, the learning of structural properties is a viable choice. For this purpose we used BN-BMLA (Antal et al., 2008b) which relies on a directed acyclic graph based Markov Chain Monte Carlo method (Madigan et al., 1996) to estimate posteriors of structural properties, such as Markov blanket graphs.

A simple approximate solution to estimate the distribution of the log odds ratio is to rely on the maximum a posteriori  $MBG(Y, G)$  with a mean parametrization learned from data and a Gaussian approximation for its variance. A more sophisticated approach takes the parametric uncertainty into account by sampling the conditional distribution parameters. Based on this sampling of parameterizations a distribution of Bayesian odds ratios arises. This allows the estimation of the credible interval of Bayesian odds ratios, which is the Bayesian analogue of the confidence interval used in the frequentist approach. The drawback of this method is that it assumes that  $MBG(Y, G)$  was identified previously or that the maximum a posteriori Markov blanket graph has such a high posterior that all others can be neglected.

A third possible approach to effect size estimation takes the structural uncertainty of  $MBG(Y, G)$  into account. This is necessary because the cardinality of Markov blanket graphs is even greater than that of Markov blanket sets, which is super exponential in the number of variables. This means that the sufficient sample size for the estimation of posteriors of Markov blanket graphs is also higher. Typically, in a practical case there are thousands of Markov blanket graphs with relatively low posteriors. Therefore, selecting the  $MBG_k(Y, G)$  with the highest posterior is not the best option. A more viable solution is to apply model averaging on Markov blanket graphs. In case of the estimation of the Bayesian odds ratio of  $X_i$  with respect to  $Y$  based on a specific  $MBG_k(Y, G)$ , the measure  $OR(X_i^{(q,r)} | X_i \in MBG_k(Y, G))$  has to be weighted according to the posterior probability of the specific Markov blanket graph  $p(MBG_k(Y, G))$ . Finally, the averaging of these

measures over possible Markov blanket graphs leads to the Bayesian MBG-based odds ratio (BOR) as

$$\begin{aligned} BOR(X_i^{(q,r)}, Y) &= \sum_{k=1}^m OR(X_i^{(q,r)} | MBG_k, \theta^{MBG_k}) \\ &\cdot p(MBG_k) \cdot 1(X_i \in MBG_k(Y, G)), \end{aligned} \quad (4)$$

where  $MBG_k$  denotes  $MBG_k(Y, G)$ ,  $m$  is the number of MBGs with a posterior  $p(MBG_k) > 0$ , and  $1()$  is the indicator function.

Even though BOR implements a fully Bayesian approach, its application is limited due to its computationally intensive nature. It can be a reasonable compromise to estimate BOR based on a selection of the  $M$  number of  $MBG_k(Y, G)$  with the highest posteriors.

## 5 Results

We investigated the properties of Bayesian odds ratio on an artificial data set (5000 samples, 115 variables) simulating a candidate gene association study. This asthma related data set contained 11 relevant variables with high posteriors of strong relevance forming the reference Markov blanket graph seen on figure 1.

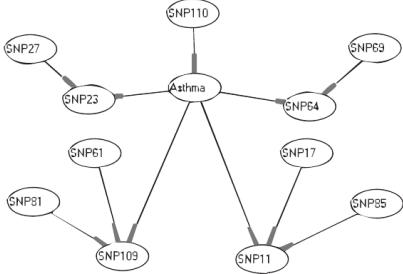


Figure 1: Reference Markov blanket graph containing all the strongly relevant variables of the data set.

The distribution of odds ratios is shown in figure 2. As depicted by figure 3 not all relevant variables had high odds ratios, in fact the majority of these relevant variables had odds ratios close to 1. A high posterior for strong relevance accompanied by an irrelevant odds ratio

can indicate that the variable is in an interaction with other variables. This is in accordance with original data set, which contained several interacting elements.

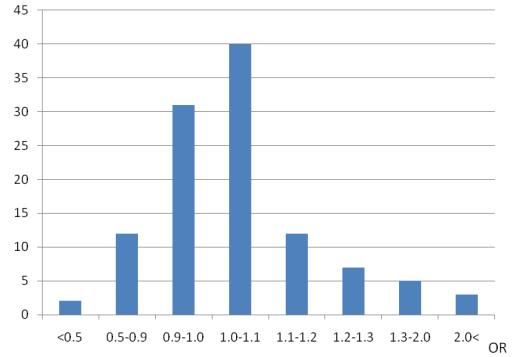


Figure 2: Distribution of odds ratios in the artificial data set.

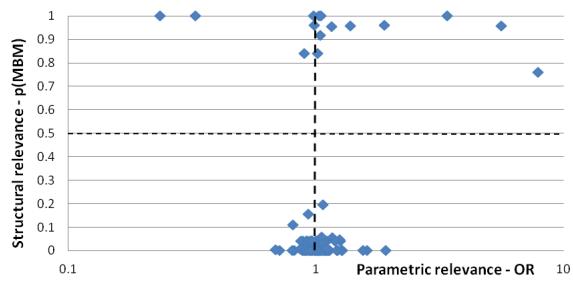


Figure 3: Structural and parametric relevance of variables of the data set. Posteriors of strong relevance are shown on the vertical, odds ratios are shown on the horizontal axis.

We computed odds ratios and corresponding confidence intervals using 300, 500, 1000 and 5000 samples. Confidence intervals were corrected for multiple hypothesis testing using the number of a priori known, strongly relevant variables (11, see figure 1). Bayesian odds ratios and related credible intervals were estimated using Markov blanket graphs learned by the BN-BMLA method. Table 1 compares the properties of two selected variables (S11 and S23) which are both strongly relevant and have high odds ratios. As figure 4 shows, in case of 5000 samples the length of Bayesian credible intervals is smaller and are contained within the cor-

rected confidence intervals in these examples.

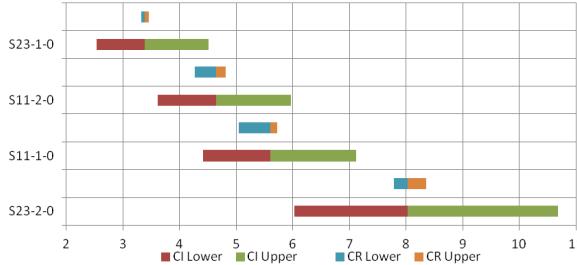


Figure 4: Comparison of confidence intervals (CI) and credible intervals (CR) of selected variables based on the data set of 5000 samples. S011-1-0 and S011-2-0 denote odds ratios of variable S011, 1 vs.0 and 2 vs.0 respectively.

With decreasing sample size the length of confidence intervals increases, whereas the length of Bayesian credible intervals remains relatively constant (see figure 5).

Table 1: Comparison of confidence intervals (CI) and Bayesian credible intervals (CR) in case of a data set of 5000 and 1000 samples. Suffixes L and U denote the Lower and the Upper half of the interval respectively.

| S-5000     | OR   | CI-L | CI-U  | CR-L | CR-U |
|------------|------|------|-------|------|------|
| S11 1 vs 0 | 5.60 | 4.41 | 7.12  | 5.05 | 5.73 |
| S11 2 vs 0 | 4.65 | 3.62 | 5.97  | 4.27 | 4.82 |
| S23 1 vs 0 | 3.38 | 2.54 | 4.51  | 3.32 | 3.46 |
| S23 2 vs 0 | 8.02 | 6.03 | 10.68 | 7.78 | 8.36 |
| S-1000     | OR   | CI-L | CI-U  | CR-L | CR-U |
| S11 1 vs 0 | 6.71 | 3.79 | 11.87 | 4.42 | 5.08 |
| S11 2 vs 0 | 5.39 | 2.96 | 9.82  | 3.65 | 4.28 |
| S23 1 vs 0 | 3.23 | 1.76 | 5.91  | 2.93 | 3.49 |
| S23 2 vs 0 | 6.87 | 3.77 | 12.53 | 6.02 | 7.56 |

However, Bayesian odds ratio has its own limitation in the small sample size domain. This is due to the insufficiency of the data for learning Markov blanket graphs. This results in remarkably different Markov blankets even among the most relevant ones. Figure 6 illustrates the case when the Bayesian odds ratio is computed separately for the 10 most relevant Markov blanket graphs for 300 samples. As it can be seen, a joint (i.e. based on all MBGs) estimation of a Bayesian credible interval is rather problematic in this case, although a possible solution is to cover the whole concerned region.

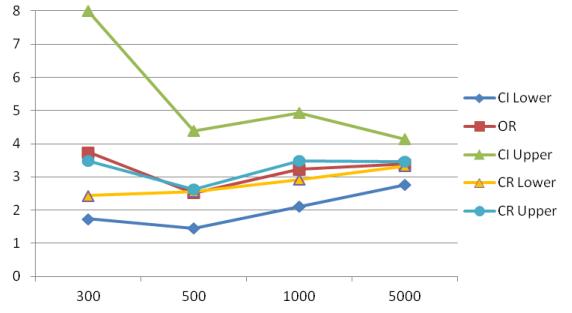


Figure 5: The effect of sample size on the confidence interval (CI) and the credible interval (CR) of a selected variable.

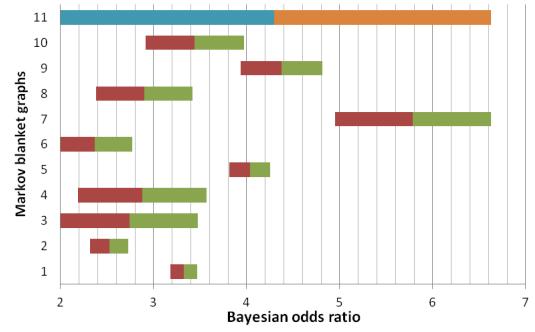


Figure 6: Bayesian Markov blanket based odds ratios of a selected variable in case of a data set of 300 samples. The vertical axis denotes the 10 highest ranking Markov blankets.

## 6 Conclusion

Bayesian networks and the Bayesian statistical framework provide multiple advantages in feature subset analysis, particularly in the exploration of interactions and in the characterization of relevance types (Ungvari et al., 2012). Beside the multivariate aspect, Bayesian networks in the Bayesian statistical framework can also be used to explore the quantitative aspect of relevance by estimating the distribution of effect size parameters. In the paper we showed that the posterior probability of strong relevance and the distribution of effect size conditioned on the strong relevance of variables provide a rich characterization of relevance. Due to the lack of analytic methods, Monte Carlo methods and approximations are

necessary. These results indicate that multiple aspects of relevance can be characterized coherently in a full Bayesian approach performing model averaging over model structures. Currently we are exploring intermediate levels between the structural (existential) and parametric (quantitative) levels, such as the monotonicity of the relation of the variables.

## Acknowledgments

This work was supported by the National Innovation Office NKTH TECH 08-A1/2-2008-0120, by the R3-COP project (Robust&Safe Mobile Co-op Syst) of ARTEMIS JU and the National Dev. Agency, and by the grant TAMOP-4.2.2.B-10/1-2010-0009.

## References

- S. Acid, L. M. de Campos, and J. G. Castellano. 2005. Learning bayesian network classifiers: searching in a space of partially directed acyclic graphs. *Machine Learning*, 59:213–235.
- P. Antal, G. Hullám, A. Gézsi, and A. Millinghoffer. 2006. Learning complex bayesian network features for classification. In *Proc. of third European Workshop on Prob. Graph. Models*, p. 9–16.
- P. Antal, A. Millinghoffer, G. Hullám, Cs. Szalai, and A. Falus. 2008a. A bayesian multilevel analysis of feature relevance. In *Workshop on New challenges for feature selection FSDM08*, p.63–74.
- P. Antal, A. Millinghoffer, G. Hullám, Cs. Szalai, and A. Falus. 2008b. A bayesian view of challenges in feature selection: Feature aggregation, multiple targets, redundancy and interaction. *JMLR Proceeding*, 4:74–89.
- D. J. Balding. 2006. A tutorial on statistical methods for population association studies. *Nature*, 7:781–91.
- W. L. Buntine. 1991. Theory refinement of Bayesian networks. In *Proc. of the 7th Conf. on Uncertainty in AI*, pages 52–60. Morgan Kaufmann.
- G. F. Cooper and E. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.
- H. Demirhan and C. Hamurkaroglu. 2008. Bayesian estimation of log odds ratios from rxc and 2x2xk contingency tables. *Statistica Neerlandica*, 62(4):405–424.
- N. Friedman and D. Koller. 2003. Being Bayesian about network structure. *Mach Learn*, 50:95–125.
- G. Hullám, P. Antal, Cs. Szalai, and A. Falus. 2010. Evaluation of a bayesian model-based approach in ga studies. *JMLR W&C Proc.*, 8:30–43.
- M. Hutter and M. Zaffalon. 2005. Distribution of mutual information from complete and incomplete data. *Comp Stat& Data An.*, 48(3):633–657.
- X. Jiang, M. M. Barmada, and S. Visweswaran. 2010. Identifying genetic interaction in genome-wide data using bayesian networks. *Genetic Epidemiology*, 34:575–581.
- R. Kohavi and G. H. John. 1997. Wrappers for feature subset selection. *AI*, 97:273–324.
- M. Koivisto and K. Sood. 2004. Exact bayesian structure discovery in bayesian networks. *Journal of Machine Learning Research*, 5:549–573.
- D. Madigan, S. A. Andersson, M. Perlman, and C. T. Volinsky. 1996. Bayesian model averaging and model selection for Markov equivalence classes of acyclic digraphs. *Comm.Statist. Theory Methods*, 25:2493–2520.
- J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, CA.
- J. Pearl. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.
- J.M. Pena, R. Nilsson, J. Björkegren, and J. Tegnér. 2007. Towards scalable and data efficient learning of markov boundaries. *International Journal of Approximate Reasoning*, 45:211–232.
- D. Rahardja, Y. D. Zhao, and H. H. Zhang. 2010. Bayesian inference of odds ratios in misclassified binary data with a validation sub-study. *Commun. Stat.-Simulation&Computation*, 39(10):1845–1854.
- M. Stephens and D.J. Balding 2009. Bayesian statistical methods for genetic association studies. *Nature Review Genetics*, 10(10):681–690.
- I. Tsamardinos and C. Aliferis. 2003. Towards principled feature selection: Relevancy,filters, and wrappers. In *Proc. of the AI&Stat.*, p. 334–342.
- I. Ungvari, G. Hullam, P. Antal, and C. Szalai et al. 2012. Evaluation of a partial genome screening of two asthma susceptibility regions using bayesian network based bayesian multilevel analysis of relevance. *PLOS ONE*, 7:1–14.
- H. Xing, P. D. McDonagh, J. Bienkowska, and J. Carulli et al. 2011. Causal modeling using network ensemble simulations of genetic and gene expression data predicts genes involved in rheumatoid arthritis. *PLoS Comp. Biol.*, 7(3):1001105.

# Latent Tree Copulas

Sergey Kirshner  
Department of Statistics  
Purdue University, USA  
[skirshne@purdue.edu](mailto:skirshne@purdue.edu)

## Abstract

We propose a new approach for estimation of joint densities for continuous observations using latent tree models for copulas, joint distributions with uniform  $\mathcal{U}(0, 1)$  marginals. Latent tree copulas combine the advantages of the parametrization of the joint density using only bivariate distributions with the ability to approximate complex dependencies with the help of latent variables. The proposed model can also be used to organize the variables in a tree hierarchy. We describe algorithms for estimating binary latent tree copulas from data for both Gaussian and non-Gaussian copulas.

## 1 Introduction

Many domains generate multivariate real-valued data (e.g., biology, finance, hydrology). Among possible approaches, non-parametric representations of the joint densities suffer from the curse of dimensionality, limiting their use to low-dimensional settings. On the other hand, while there are many parametric forms for one-dimensional densities, the choices for multivariate densities with desirable properties (computationally convenient functional forms, easy estimation of marginal, conditional, and posterior densities) are limited, with the best known, understood, and used tool, multivariate normal distributions, not always applicable for the domain. The latter issue has been one of the reasons why probabilistic graphical models (PGMs) have been slow to extend to non-Gaussian cases.

*Copulas*, multivariate distributions with uniform  $\mathcal{U}(0, 1)$  marginals (e.g., Joe, 1997, Nelsen, 2006), provide a convenient framework for modeling of multivariate distributions as this task can be split into two parts: (1) modeling of the univariate marginal distributions, and (2) modeling of the copula which would “bind” the univariate marginals together to make up a joint distribution. This approach provides flexibility of separate specification of the functional form

for the copula and the marginals. Copulas can be viewed as a canonical representation for the dependence between the random variables as they preserve the conditional independence relations between the variables while fixing their marginals. This motivates the use of PGMs for the copulas rather than the joint densities, and recent efforts have been extending PGM techniques to copulas (e.g., Elidan, 2010a,b, Kirshner, 2007).

In this paper, we propose *latent tree copulas* (LTCs), tree-structured copulas with some of the variables hidden (not observed), for modeling of multivariate densities. This approach combines two frameworks, tree-structured copulas (Kirshner, 2007) and latent tree models (Zhang, 2004), borrowing strengths from both. Tree-structured copulas decompose the joint copula density into a product of bivariate copula densities (corresponding to the edges in the acyclic conditional independence graph) allowing to use the existing bivariate copula machinery for construction of high-dimensional copulas. The requirement of only bivariate copulas is important because while the choices for bivariate copulas have been thoroughly investigated, only relatively few bivariate functional families have useful multivariate generalizations. However, the conditional independence assumptions imposed by the tree-structure are often unreal-

istic. Borrowing an idea from latent tree models and introducing latent variables allows modeling of more complex dependence relations.

This paper contributes a new parsimonious model for multivariate data and algorithms for its inference and parameter estimation. We also propose a learning algorithm (in the vein of Harmeling and Williams, 2011) to estimate binary latent tree copulas from data. The learned structure can be used for hierarchical clustering of the variables or features.<sup>1</sup>

First, we place the contributions of this paper in the context of related work (Section 1.1). We then introduce copulas, and a particular tree-structured subclass of them (Section 2). This provides enough background to introduce the latent copula trees (Section 3). Estimation of latent copula trees from data is considered in Section 4. Empirical illustration of the above approaches is carried out in Section 5. We conclude in Section 6.

## 1.1 Related Work

So far, the approaches for learning of latent tree models focus mostly on the discrete random variables (Harmeling and Williams, 2011, Zhang, 2004, Zhang and Kočka, 2004) although recent work of Choi et al. (2011) considered the jointly normal case in addition to discrete variables. Copulas (e.g., Joe, 1997, Nelsen, 2006) are becoming an important tool for dealing with non-Gaussian data with applications in many areas, e.g., finance (Cherubini et al., 2004), Hydrology (Genest and Favre, 2007), and they are attracting the attention of the machine learning community.<sup>2</sup> Several graphical models have been considered for copulas including trees (Kirshner, 2007) which decompose the joint copula density into a product of bivariate copula densities, and several of their generalizations, among them, vines which incorporate additional conditional dependence with nested bivariate copulas (Aas et al., 2009, Bedford and

<sup>1</sup>An implementation of the algorithms described in the paper is available for download from <http://www.stat.purdue.edu/~skirshne/LTC/>.

<sup>2</sup>NIPS 2011 workshop on Copulas in Machine Learning, December 2011, <http://pluto.huji.ac.il/~galelidan/CopulaWorkshop/abstracts.html>.

Cooke, 2002, Kurowicka and Cooke, 2006), and copula Bayesian networks (Elidan, 2010a) which decompose the joint copula density into a product of conditional copula densities and can handle missing data using variational approach (Elidan, 2010b) (although the case of *latent* variables was not explored). A related (albeit not a copula) model is a cumulative distribution network (Huang and Frey, 2008) which models the distribution function as a product of distribution function factors. Its recent extension, a mixed cumulative distribution network (Silva et al., 2011) uses implicit hidden variables.

## 2 Tree-Structured Copulas

Let  $\mathbf{X} = (X_1, \dots, X_d)$  be a vector of random variables with support  $Val(\mathbf{X}) \subseteq \mathbb{R}^d$ . Let  $F(\mathbf{x}) = F(\mathbf{X} = \mathbf{x})$  be a cumulative distribution function (cdf) for  $\mathbf{X}$ , and assume that  $F$  is absolutely continuous. Denote by  $f(\mathbf{x})$  the probability density function (pdf) for  $\mathbf{X}$ . Let  $F_u(x_u)$  and  $f_u(x_u)$ ,  $u = 1, \dots, d$ , denote the marginal cdfs and pdfs, respectively, of  $\mathbf{X}$ .

### 2.1 Copulas

Copulas provide a convenient framework for modeling of multivariate distributions by separating the marginals from the multivariate dependence. Let  $a_u = F_u(x_u)$  be the marginal distribution function for variable  $X_u$ , and let  $\mathbf{a} = (a_1, \dots, a_d)$ . Denote by  $\mathbb{I} = (0, 1)$  the unit interval. A *copula* associated with  $F$  is a distribution function (cdf)  $C : \mathbb{I}^d \rightarrow \mathbb{I}$  satisfying

$$F(\mathbf{x}) = C(F_1(x_1), \dots, F_d(x_d)), \quad \mathbf{x} \in \mathbb{R}^d,$$

and if  $F$  is absolutely continuous on  $\mathbb{R}^d$ ,  $C(\mathbf{a}) = F(F_1^{-1}(a_1), \dots, F_d^{-1}(a_d))$ , and such decomposition of a distribution into its marginals and its copula is unique (Sklar’s Theorem, Sklar, 1959).<sup>3</sup> For the absolutely continuous case, the probability density function  $f$  can be represented in terms of the marginal densities

<sup>3</sup>If  $F$  is not absolutely continuous,  $C$  can be obtained using the generalized inverse of the marginal cdfs, and the associated with  $F$  copula function is uniquely defined on the absolutely continuous region of the support for  $\mathbf{X}$ .

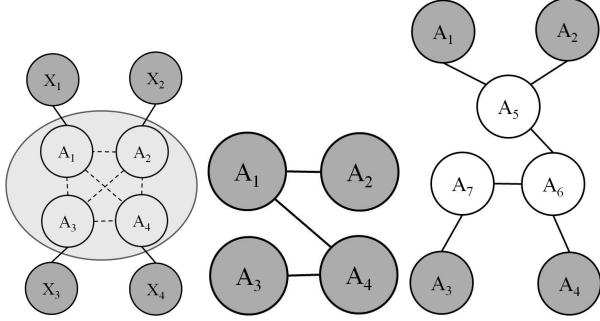


Figure 1: Left: graphical model for copulas. Copula variables are  $A_1, \dots, A_4$  with unknown dependence; original variables are  $X_1, \dots, X_4$  (observed). Middle: tree-structured copula model for  $A_1, \dots, A_4$ . Right: latent tree copula model for  $A_1, \dots, A_4$ .

$f_1, \dots, f_d$  and the *copula density function*  $c$ :

$$f(\mathbf{x}) = c(\mathbf{a}) \prod_{u=1}^d f_u(x_u), \quad c(\mathbf{a}) = \frac{\partial^d C(\mathbf{a})}{\partial a_1 \dots \partial a_d}. \quad (1)$$

Thus a multivariate distribution can be constructed by choosing univariate marginals  $F_1, \dots, F_d$ , and then coupling or “gluing” them together with a *separately* chosen multivariate distribution, copula  $C$ . Further, the product decomposition in (1) suggests a convenient parameter estimation approach for a multivariate distribution by first estimating the marginals, transforming individual components independently according to the estimated marginals, and then estimating the parameters of the copula based on the transformed values. For a graphical model of a copula see Figure 1 (left).

For in-depth treatment of copulas, please see e.g., Joe (1997), Nelsen (2006). However, we will introduce the Gaussian copula family as it has a number of useful properties discussed in this paper. Suppose  $(X_1, X_2)^T$  is a vector of two jointly Gaussian random variables (r.v.s) with mean  $(\mu_1, \mu_2)^T$  and covariance matrix  $\begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$ . The copula associated with the distribution over  $(X_1, X_2)$  is

$$C(a_1, a_2) = \Phi_\rho \left( \Phi^{-1} \left( \frac{x_1 - \mu_1}{\sigma_1} \right), \Phi^{-1} \left( \frac{x_2 - \mu_2}{\sigma_2} \right) \right)$$

where  $\Phi$  is the standard normal cdf, and  $\Phi_\rho$  is the bivariate normal cdf for the pair of standard normal r.v.s with correlation  $\rho \in [-1, 1]$ . Gaussian copula easily generalizes to  $d > 2$  by replacing  $\rho$  with a correlation matrix  $R$ . Gaussian copula family (bivariate or multivariate) is perhaps the most commonly employed as it shares many properties with Gaussian distributions.

## 2.2 Copulas with a Markov Tree Model

In this paper, we are focusing on the *tree-structured* PGMs. Suppose a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is an undirected tree with the set of nodes  $\mathcal{V} = \{1, \dots, d\}$ , and assumes that  $\mathcal{G}$  contains no cycles. Assuming  $X_1, \dots, X_d$  satisfy the Markov assumptions encoded by  $\mathcal{G}$ , the joint pdf  $f$  can be written as

$$f(\mathbf{x}) = \left[ \prod_{u=1}^d f_u(x_u) \right] \left[ \prod_{\{u,v\} \in \mathcal{E}} \frac{f_{uv}(x_u, x_v)}{f_u(x_u) f_v(x_v)} \right] \quad (2)$$

where  $f_{uv}(x_u, x_v)$  denotes the bivariate marginal density for  $(X_u, X_v)$ . The same product decomposition holds for the case of discrete random variables  $\mathbf{X}$  with probability mass functions (pmfs) replacing pdfs.

Kirshner (2007) proposed using tree-structured copulas and their variants for multidimensional density estimation (Figure 1, middle). By combining (2) and (1), the copula density for a tree-structured distribution can be expressed as a product of bivariate copulas on the edges:

$$c_T(\mathbf{a}) = \prod_{\{u,v\} \in \mathcal{E}} c_{uv}(a_u, a_v). \quad (3)$$

The converse also holds; if a pdf  $c_T(\mathbf{a})$  is constructed as a product of bivariate copulas as in (3), then it is a valid copula density. This property permits building high-dimensional tree-structured copulas by separately specifying the Markov tree-structure and a bivariate copula (or its densities) for each edge.

## 3 Latent Tree Copulas

Copula decomposition suggests that the integral part of modeling the multivariate densities

is modeling their copulas. While computationally convenient, it is unreasonable to expect the multivariate dependence to be tree-structured. Mixtures or ensembles of trees can model more complex dependencies (Kirshner, 2007, Meilă and Jordan, 2000), but they may lack interpretability. Our approach extends the latent tree model of Zhang (2004) to copulas and thus to real-valued data by introducing latent variables (LVs) for copula models while preserving the appealing properties of trees.

**Definition 1.** Let  $\mathbf{A} = (A_1, \dots, A_d)^T$  be a vector of  $\mathcal{U}(0, 1)$  random variables.  $\mathbf{A}$  is *tree-decomposable* if there exists a tree (forest)  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ,  $\mathcal{V} = \{1, \dots, t\}$ ,  $t \geq d$ , and bivariate copula densities  $c_{uv}(a_u, a_v)$ ,  $\{u, v\} \in \mathcal{E}$  such that

$$c_{LT}(\mathbf{a}) = \iint_{\mathbb{R}^{t-d}} \prod_{\{u,v\} \in \mathcal{E}} c_{uv}(a_u, a_v) da_{d+1} \dots da_t. \quad (4)$$

We call the copula  $C_{LT}$  in (4) a *latent tree copula* (LTC).

LTCs generalize the tree-structured copulas by introducing LVs  $A_{d+1}, \dots, A_t$ . Unlike the categorical variable setting where the support for each variable is finite, and therefore, bivariate marginals can be naturally represented with a multinomial distribution, modeling of  $c_{uv}(a_u, a_v)$  requires additional assumptions. We assume the density for each  $c_{uv}$  is parametric with a pair  $(\mathcal{M}_{uv}, \theta_{uv})$  denoting the functional form and a vector of parameters, respectively. To define a LTC model for  $(A_1, \dots, A_d)$ , one needs to specify a 4-tuple  $(t, \mathcal{E}, \mathcal{M}, \boldsymbol{\theta})$ :  $t$ , the total number of variables,  $\mathcal{E}$ , the set of  $t - 1$  index pairs,  $\mathcal{M} = (\mathcal{M}_{uv})_{\{u,v\} \in \mathcal{E}}$ , the set of  $t - 1$  copula functional forms, and  $\boldsymbol{\theta} = (\theta_{uv})_{\{u,v\} \in \mathcal{E}}$ , the set of  $t - 1$  vectors of parameters for the bivariate copulas.

Using LVs within the trees permits approximating densities, possibly with complex dependencies, using only bivariate copulas. As is the case with tree copulas, this opens up a significant body of existing work on bivariate copulas for construction of multivariate densities, in contrast to copula Bayesian networks (Elidan, 2010a) which require higher-dimensional copu-

las as building blocks. On the other hand, in contrast to mixtures or ensembles of trees, the latent tree copulas provide a clear interpretation of the dependence between the variables in the model.

LTCs have a somewhat different representational power than their categorical-valued “siblings”, LTMs (Zhang, 2004). LTMs are distributions over a finite set of variable states and are not identifiable as many different LTMs can represent the same distribution over the observed variables (OVs)  $(X_1, \dots, X_d)$ , (property known as a marginal equivalence of the models). In order to evaluate LTMs, it is therefore necessary to consider the *parsimony* of the model, with preference given to models with fewer free parameters. However, among parsimonious families of LTMs, there are only a finite number of tree-structures possible to represent all possible distributions over a fixed number of OVs (Zhang, 2004). LTCs, depending on the functional families for copulas,  $\mathcal{M}$ , could have a very large number of parameters and cannot be represented by a simpler model. For practical purposes, we have to limit the number of latent variables in the LTC model, and we have to consider bivariate copula families which use few parameters.

## 4 Learning

Since LTC is specified as a 4-tuple,  $(t, \mathcal{E}, \mathcal{M}, \boldsymbol{\theta})$ , potentially all 4 elements of the 4-tuple need to be estimated. In the categorical case, the number of possible minimal models agreeing with marginals over the OVs is superexponential in  $d$ , but is finite (Zhang, 2004). For the continuous case, however, there is a potentially infinite number of models which can approximate the data (as for most copula families for the edges, adding additional edges simply increases the flexibility of the model).

To make learning tractable, we therefore have to restrict the class of desired solutions. We make two assumptions: (1) we assume that each bivariate copula comes from the same functional family,  $\mathcal{M}_{uv} = \mathcal{M}'$  for all  $\{u, v\} \in \mathcal{E}$  with  $\mathcal{M}'$  selected apriori, and (2) we restrict the struc-

tures  $\mathcal{G}$  to be binary latent trees as in Harmeling and Williams (2011). Binary latent tree with  $d > 1$  observed (manifest) variables has  $d - 1$  LVs ( $t = 2d - 1$ ); each observed variable corresponds to a leaf of the tree, and each latent variable has exactly 3 neighbor except for one latent variable, the root of the tree, see Figure 1, right ( $d = 4$ ).

**Theorem 1.** *Any Gaussian LTC can be represented as a Gaussian binary LTC.<sup>4</sup>*

Thus it is justifiable to consider only binary LTCs for the Gaussian case even though Gaussian binary LTCs will have redundant latent nodes, and potentially could have edges joining perfectly dependent variables (i.e.,  $\theta_{uv} \in \{-1, 1\}$ ). For the non-Gaussian case, binary latent trees may not necessarily represent all possible tree-decomposable distributions within the family. Still, the families of binary latent trees are flexible, and the resulting trees may provide an intuitive hierarchical interpretation for the components of  $\mathbf{A}$ .

#### 4.1 Structure and Parameter Estimation

For estimation of binary latent trees and their parameters, we propose a greedy algorithm **Greedy-BLTC**, Algorithm 1, which repeatedly merges latent binary trees over subsets of variables into larger subtrees by introducing new latent variables as a root. The algorithm is an adaptation of **(Bin-G)** from Harmeling and Williams (2011) with some modifications to LTCs. The algorithm starts out with each variable  $A_1, \dots, A_d$  as the root of its own tree, with the joint distribution with the *working set* denoting the set of current roots. The variables in different trees are independent; the joint probability distribution over  $\mathbf{A}$  is the product of probability distributions for each tree. At each step, the two trees with the highest estimated mutual information (based on the posterior probabilities for the latent variables) between their roots variables are merged into a larger tree by creating a new latent variable as the root of the new

<sup>4</sup>Proof is omitted due to space limitations and will appear in the full version.

---

#### Algorithm 1 Greedy-BLTC

---

```

1: input: a working set $V = \{1, \dots, d\}$ of indices for
 variables A_1, \dots, A_d
2: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = V$, $\mathcal{E} = \emptyset$
3: estimate pairwise MI for each pair of variables
4: for $r = d + 1, \dots, 2d - 1$ do
5: /* loop over the index of the new LV (new root) */
6: $\{u, v\} \leftarrow$ pair from V with highest MI for
 (A_u, A_v)
7: remove u and v from V
8: add new root r to V /* variable A_r is latent */
9: add r to \mathcal{V} and edges $\{r, u\}$ and $\{r, v\}$ to \mathcal{E}
10: $\boldsymbol{\theta} \leftarrow \text{EstimateParameters}(\text{subtree with root } r)$
 using the EM algorithm (see Section 4.2)
11: estimate pairwise MIs between r and the rest of
 working set V
12: end for
13: output: the graph \mathcal{G} , with $\mathcal{V} = \{1, \dots, 2d - 1\}$,
 parameters $\boldsymbol{\theta} = \{\theta_{uv}\}_{\{u,v\} \in \mathcal{E}}$

```

---

tree and joining the new variable to the roots of the two subtrees. The parameters of the newly created tree are re-estimated using the EM algorithm (Section 4.2). The process is repeated until all of the nodes belong to the same tree.

#### 4.2 Parameter Estimation Given Structure

To estimate the parameters of a subtree from line 10 of Algorithm 1, we will employ a variant of the EM algorithm (Dempster et al., 1977). Suppose there are  $d$  observed variables labeled  $\mathbf{A}_O = (A_1, \dots, A_d)$  and  $t - d$  latent variables labeled  $\mathbf{A}_H = (A_{d+1}, \dots, A_t)$ ,  $\mathcal{M}'$  is given, and suppose  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is a tree. (The tree does not have to be binary.) Suppose we are given a set of i.i.d. observation vectors  $\mathcal{D} = \{\mathbf{a}_O^1, \dots, \mathbf{a}_O^N\}$ ,  $\mathbf{a}_O^n = (a_1^n, \dots, a_d^n)$  with none of  $a_u^n$  ( $u = 1, \dots, d$ ) missing. Our goal is to find the maximum likelihood estimate (MLE) of the parameters  $\boldsymbol{\theta} = \{\theta_{uv}\}_{\{u,v\} \in \mathcal{E}}$ :

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^N \ln c_{LT}(\mathbf{a}_O^n | \boldsymbol{\theta}). \quad (5)$$

Each iteration of the standard EM optimizes

$$\sum_{n=1}^N E_{c_{LT}(A_u^n, A_v^n | \mathbf{a}_O^n, \boldsymbol{\theta}') \ln c_{uv}(a_u^n, a_v^n | \theta_{uv})} \quad (6)$$

with respect to  $\theta_{uv}$  for each  $\{u, v\} \in \mathcal{E}$ , where  $\boldsymbol{\theta}'$  is the set of parameter values at the start of

the iteration. For the  $\mathcal{M}' = \text{Gaussian}$  family of copulas, the update can be computed in closed form by finding a root  $\theta_{uv} \in [-1, 1]$  of the equation

$$\alpha_3\theta_{uv}^3 + \alpha_2\theta_{uv}^2 + \alpha_1\theta_{uv} + \alpha_0 = 0$$

where

$$\begin{aligned}\alpha_3 &= N, \\ \alpha_2 &= \alpha_0 = -\sum_{n=1}^N E_{\mathcal{N}_T(Z_u^n, Z_v^n | \mathbf{z}^n, \boldsymbol{\theta}')} z_u^n z_v^n, \\ \alpha_1 &= \sum_{n=1}^N E_{\mathcal{N}_T(Z_u^n, Z_v^n | \mathbf{z}^n, \boldsymbol{\theta}')} \left[ (z_u^n)^2 + (z_v^n)^2 - 1 \right].\end{aligned}$$

$\mathcal{N}_T(\mathbf{Z}^n | \boldsymbol{\theta}')$  is a tree-structured  $t$ -variate Gaussian distribution obtained from  $C_{LTN}(\mathbf{A}^n | \boldsymbol{\theta}')$  via independent transformation of the marginals  $Z_u^n = \Phi^{-1}(A_u^n)$ . The expectations above can be efficiently computed using the Belief Propagation algorithm of Pearl (1988) applied to a multivariate normal  $\mathcal{N}_T$ ,  $\mathcal{O}(Nt)$  computational complexity.

#### 4.2.1 Non-Gaussian Case

For cases other than  $\mathcal{M}' = \text{Gaussian}$ , the posterior copula density  $c_{LT}(Z_u^n, Z_v^n | \mathbf{z}^n, \boldsymbol{\theta}')$  may not be computable in closed form, and for these families a direct application of EM is therefore impossible. Instead, we propose using a variational approach. For any density  $q^n(\mathbf{a}_H^n)$  over  $\mathbb{I}^{t-d}$ ,

$$\begin{aligned}&\ln c_{LT}(\mathbf{a}_O^n | \boldsymbol{\theta}') \\ &= \int_{\mathbb{I}^{t-d}} q^n(\mathbf{a}_H^n) \ln \frac{c_{LT}(\mathbf{a}_O^n, \mathbf{a}_H^n | \boldsymbol{\theta}')}{q^n(\mathbf{a}_H^n)} d\mathbf{a}_H^n \quad (7) \\ &+ D(q^n(\mathbf{a}_H^n) \| c_{LT}(\mathbf{a}_H^n | \mathbf{a}_O^n, \boldsymbol{\theta}')).\end{aligned}$$

For variational EM (e.g., Wainwright and Jordan, 2008), the family  $\mathcal{Q}$  of distributions  $q^n \in \mathcal{Q}$  is chosen in a way so that  $D(q^n \| c_{LT}(\cdot | \mathbf{a}_O^n, \boldsymbol{\theta}'))$  can be easily minimized. Notice that unlike the standard setting for variational inference on graphical models, the approximation is not needed to simplify the dependence structure of latent variables conditioned on the observations; the dependence structure imposed on  $c_{LT}(\mathbf{a}_H^n | \mathbf{a}_O^n)$  by  $\mathcal{E}$  is already a tree or a forest. Instead for LTCs,

the approximating family  $\mathcal{Q}$  needs to be chosen so that inference with it does not require closed form marginalization of copula functions or their products. The modified EM algorithm iterates between choosing  $q^n(\mathbf{a}_H^n)$  minimizing  $D(q^n \| c_{LT}(\cdot | \mathbf{a}_O^n, \boldsymbol{\theta}'))$ ,  $n = 1, \dots, N$ , (E-step) and updating parameters  $\boldsymbol{\theta} = \{\theta_{ij}\}_{\{i,j\} \in \mathcal{E}}$  (M-step):

$$\hat{\theta}_{uv} = \underset{\theta_{uv}}{\operatorname{argmax}} \sum_{n=1}^N E_{q_{uv}^n(\mathbf{A}_H^n)} \ln c_{uv}(a_u^n, a_v^n | \theta_{uv}) \quad (8)$$

Let  $\mathcal{E}_H = \{\{u, v\} \in \mathcal{E} : u, v \in H\}$  be the subset of edges joining only the latent variables, and let  $\mathcal{E}_{H+} = \{\{u, v\} \in \mathcal{E} : u \in H, v \in O\}$  be the subset of edges joining one latent and one observed variable. We consider a family of tree structured distributions  $\mathcal{Q}$  with Markov graph  $\mathcal{G}_H = (H, \mathcal{E}_H)$  so that  $\forall q \in \mathcal{Q}$

$$q(\mathbf{a}_H) = \prod_{u \in H} q_u(a_u) \left[ \prod_{\{u,v\} \in \mathcal{E}_H} \frac{q_{uv}(a_u, a_v)}{q_u(a_u) q_v(a_v)} \right]$$

where  $q_{uv}$ ,  $q_u$ , and  $q_v$  are marginals of  $q$  for the variables  $(A_u, A_v)$ ,  $A_u$ , and  $A_v$ , respectively. We represent a pdf  $q_{uv}(a_u, a_v)$  on  $\mathbb{I}^2$  as a 2-d step function partitioning  $\mathbb{I}^2$  into  $K^2 \frac{1}{K} \times \frac{1}{K}$  squares (for some preselected integer  $K$ ), with each square having uniform density  $p_{uv}(i, j)$ :

$$\begin{aligned}q_{uv}(a_u, a_v) &= p_{uv}(i, j) \geq 0 \text{ for } a_u \in \mathbb{I}_i, a_v \in \mathbb{I}_j, \\ q_u(a_u) &= p_u(i) \geq 0 \text{ for } a_u \in \mathbb{I}_i, \\ \text{where } \mathbb{I}_i &= \left( \frac{i-1}{K}, \frac{i}{K} \right], \text{ and} \\ \sum_{i=1}^K p_u(i) &= K, \quad u \in H, \quad i = 1, \dots, K, \\ \sum_{i=1}^K p_{uv}(i, j) &= K p_v(j), \quad \text{and} \\ \sum_{j=1}^K p_{uv}(i, j) &= K p_u(i), \quad \forall \{u, v\} \in \mathcal{E}_H \quad (9)\end{aligned}$$

with (9) ensuring  $q$  has proper marginals, and that the bivariate densities agree on the marginals.

Minimizing  $D(q^n \| c_{LT}(\cdot | \mathbf{a}_O^n))$  is equivalent to maximizing  $\int_{\mathbb{I}^{t-d}} q^n(\mathbf{a}_H^n) \ln \frac{c_{LT}(\mathbf{a}_O^n, \mathbf{a}_H^n | \boldsymbol{\theta}')}{q^n(\mathbf{a}_H^n)} d\mathbf{a}_H^n$

(Eqn 7), and is equivalent to minimizing

$$\begin{aligned}
f(q^n) &= \frac{1}{K} \sum_{u \in H} \sum_{i=1}^K p_u^n(i) \ln p_u^n(i) \\
&+ \frac{1}{K^2} \sum_{\{u,v\} \in \mathcal{E}_H} \sum_{i=1}^K \sum_{j=1}^K p_{uv}^n(i,j) \ln \frac{p_{uv}^n(i,j)}{p_u^n(i)p_v^n(j)} \\
&- \sum_{\{u,v\} \in \mathcal{E}_H^+} \sum_{i=1}^K w_u^n(i) p_u^n(i) \\
&- \sum_{\{u,v\} \in \mathcal{E}_H} \sum_{i=1}^K \sum_{j=1}^K w_{uv}^n(i,j) p_{uv}^n(i,j), \text{ where}
\end{aligned}$$

$$\begin{aligned}
w_u^n(i) &= \int_{\mathbb{I}_i} \ln c_{uv}(a_u^n, a_v^n) da_u^n, \\
w_{uv}^n(i,j) &= \int_{\mathbb{I}_i} \int_{\mathbb{I}_j} \ln c_{uv}(a_u^n, a_v^n) da_u^n da_v^n.
\end{aligned} \tag{10}$$

Whether mean field approximation is used (i.e., assuming  $p_{uv}^n(i,j) = p_u^n(i)p_v^n(j)$ ) or structured mean field, it is straightforward to derive a set of fixed point equations to minimize  $f(q^n)$  subject to the constraints in (9). The integrals of log-copula densities (10) do not have analytic expressions for most bivariate copula families. We employ quadrature methods for estimation of these integrals as they are low-dimensional and have bounded range of integration. While the log-likelihood cannot be evaluated directly, it can be lower-bounded (from Eqn 7), by  $-\sum_{n=1}^N f(q^n)$ . In our experiments, choosing  $K \geq 50$  led to good fits of  $q^n$  to  $c_{uv}(a_H^n | a_O^n, \theta'_{uv})$  and thus to a good approximation of the log-likelihood in the equation above.

The computational complexity for the proposed approach depends on the number of iterations until mean field equations converge. Per update, the complexity is  $\mathcal{O}(dK^2)$  per data point. This does not include the complexity of computing the integrals in (10),  $Nd$  univariate and  $d - 1$  bivariate such integrals for latent binary trees.

## 5 Experimental Illustration

For an illustration, we model the S&P 100 monthly stock returns data set described in

Table 1: Comparison of the log-likelihood, BIC, number of created latent variables, number of free parameters, and running time for the S&P 100 monthly stock returns data.

|             | ll    | #latent | xval8  |
|-------------|-------|---------|--------|
| LTC-G       | 25381 | 84      | 108.38 |
| CL          | 23970 | 0       | 105.99 |
| NJ          | 24408 | 45      | 108.45 |
| CLRJ        | 24361 | 26      | 108.50 |
| Copula-CL   | 24787 | 0       | 107.96 |
| Copula-NJ   | 25350 | 41      | 110.08 |
| Copula-CLRJ | 25284 | 30      | 109.68 |

Choi et al. (2011) with a Gaussian LTC.<sup>5</sup> The data set consists of 216 monthly stock returns of 84 companies in the S&P 100 stock index (and the index itself) for the years 1990–2007. The goal is to approximate the high-dimensional distribution between the returns and to discover useful hierarchies among the variables in question. First, we transformed the data into the copula domain: the marginal densities  $f_u$ ,  $u = 1, \dots, 85$  were estimated by Gaussian KDEs with bandwidths determined using the Rule of Thumb (Silverman, 1986), and the data was mapped into  $\mathbb{I}^d$  ( $d = 85$ ) by applying  $F_u$  to each component  $u$  of the data vector.<sup>6</sup> We then fit a Gaussian LTC (LTC-G) trained using Greedy-BLTC (Algorithm 1) to the transformed data using 10 random restarts for parameter estimation (EM) within each subtree. The results are listed in Table 1; the likelihoods and the number of hidden variables are computed for the original (not transformed) data; xval8 refers to out-of-sample per-example log-likelihood obtained by 8-fold cross-validation. CL, NJ, and CLRJ procedures are described in Choi et al. (2011); the goal of all of these approaches is to learn a latent tree Gaussian model. Copula- CL, NJ, and CLRJ differ in

<sup>5</sup>The data set is available as a part of the software toolbox for Choi et al. (2011). <http://people.csail.mit.edu/myungjin/latentTree.html>

<sup>6</sup>Non-parametric estimation of marginal distributions is a common approach in copula modeling (e.g., Joe and Xu, 1996).

that their marginals are first mapped into a copula domain, and are then modified by the inverse normal CDF transform. However, the marginals in this case are close to normal, and the improvement of the copula- versions of the algorithms is not significant. LTC-G appears to provide a similar fit as suggested by the log-likelihood.<sup>7</sup>

The graph displaying the hierarchy of the variables is omitted due to limited space. However, similar to the reports in Choi et al. (2011), our approach generates interpretable substructures. For example, there is a subtree populated entirely by the natural gas and oil production and exploration companies (Schlumberger, Baker Hughes, Halliburton, Occidental Petroleum, Exxon, Chevron, ConocoPhillips), and another by the telecommunication companies (Spring, Verizon, and AT&T).

## 6 Conclusion

We proposed a new model for multivariate continuous data based on a latent tree hierarchy for the copula of its joint distribution. This model can be used both to model high-dimensional densities without the jointly Gaussian assumption or to group variables into subgroups. We described an algorithm for estimation of the model's binary tree structure together with its parameters from data. In the future, we plan to improve the estimation procedure for the non-Gaussian copula case. We are planning to use the above model for problems in hydrology, in particular, as an approach to regionalization of watersheds.

## Acknowledgments

This work has been supported by the US National Science Foundation award AGS-1025430.

## References

- K. Aas, C. Czado, A. Frigessi, and H. Bakken. Pair-copula constructions of multiple dependence. *Ins.: Mathematics Econ.*, 44(2):182 – 198, 2009.
- T. Bedford and R. M. Cooke. Vines – a new graphical model for dependent random variables. *Ann. Stat.*, 30(4):1031–1068, 2002.
- U. Cherubini, E. Luciano, and W. Vecchiato. *Copula Methods in Finance*. Wiley, 2004.
- M. J. Choi, V. Y. Tan, A. Anandkumar, and A. S. Wilksky. Learning latent tree graphical models. *JMLR*, 12:1771–1812, May 2011.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via EM algorithm. *J. Roy. Statist. Soc. Ser. B*, 39(1):1–38, 1977.
- G. Elidan. Copula bayesian networks. In *NIPS*, pages 559–567, 2010a.
- G. Elidan. Inference-less density estimation using copula bayesian networks. In *UAI*, pages 151–159, 2010b.
- C. Genest and A.-C. Favre. Everything you always wanted to know about copula modeling but were afraid to ask. *Journal of Hydrological Engineering*, 12(4):347–368, July 2007.
- S. Harmeling and C. K. I. Williams. Greedy learning of binary latent trees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(6):1087–1097, June 2011.
- J. C. Huang and B. J. Frey. Cumulative distribution networks and the derivative-sum-product algorithm. In *UAI*, pages 290–297, 2008.
- H. Joe. *Multivariate Models and Dependence Concepts*. Chapman & Hall/CRC, 1997.
- H. Joe and J. J. Xu. The estimation method of inference functions for margins for multivariate models. Technical report, Department of Statistics, University of British Columbia, 1996.
- S. Kirshner. Learning with tree-averaged densities and distributions. In *NIPS*, pages 761–768, 2007.
- D. Kurowicka and R. M. Cooke. *Uncertainty Analysis with High Dimensional Dependence Modelling*. Wiley, 2006.
- M. Meilă and M. I. Jordan. Learning with mixtures of trees. *JMLR*, 1(1):1–48, October 2000.
- R. B. Nelsen. *An Introduction to Copulas*. Springer, 2nd edition, 2006.
- J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.
- R. Silva, C. Blundell, and Y. W. Teh. Mixed cumulative distribution networks. *JMLR - Proceedings Track (AISTATS-2010)*, 15:670–678, 2011.
- B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
- A. Sklar. Fonctions de repartition à  $n$  dimensions et leurs marges. *Publications de l'Institut de Statistique de L'Université de Paris*, 8:229–231, 1959.
- M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.
- N. L. Zhang. Hierarchical latent class models for cluster analysis. *JMLR*, 5:697–723, June 2004.
- N. L. Zhang and T. Kočka. Efficient learning of hierarchical latent class models. In *ICTAI*, pages 585–593, 2004.

<sup>7</sup>RG and CLNG from Choi et al. (2011) and their copula versions performed worse than NJ and CLRJ.

# Learning Mixtures of Truncated Basis Functions from Data

Helge Langseth

Department of Computer and Information Science  
The Norwegian University of Science and Technology  
Trondheim (Norway)  
[helgel@idi.ntnu.no](mailto:helgel@idi.ntnu.no)

Thomas Dyhre Nielsen

Department of Computer Science  
Aalborg University  
Aalborg (Denmark)  
[tdn@cs.aau.dk](mailto:tdn@cs.aau.dk)

Antonio Salmerón

Department of Statistics and Applied Mathematics  
University of Almería, Almería (Spain)  
[antonio.salmeron@ual.es](mailto:antonio.salmeron@ual.es)

## Abstract

In this paper we describe a new method for learning hybrid Bayesian network models from data. The method utilizes a kernel density estimator, which is in turn “translated” into a *mixture of truncated basis functions*-representation using a convex optimization technique. We argue that these estimators approximate the maximum likelihood estimators, and compare our approach to previous attempts at learning hybrid Bayesian networks from data. We conclude that while the present method produces estimators that are slightly poorer than the state of the art (in terms of log likelihood), it is significantly faster.

## 1 Introduction

In domains involving both discrete and continuous variables, Bayesian networks with mixtures of truncated exponentials (MTEs) (Moral et al., 2001) and mixtures of truncated polynomials (MOPs) (Shenoy and West, 2011) have received increasing interest over the last few years. A recent addition to the fold is the *mixtures of truncated basis functions* (MoTBFs) framework (Langseth et al., 2012), which offers a unified theory for MTEs and MoPs. The MoTBFs framework allows discrete and continuous variables to co-exist in a Bayesian network without any structural constraints, and since the family of MoTBFs is closed under addition, multiplication, and integration, inference in an MoTBF network can be performed efficiently using the Shafer-Shenoy architecture (Shafer and Shenoy, 1990).

The problem of learning MoTBF models from data has been only scarcely considered, with the main body of work relating to MTEs (Romero et al., 2006; Langseth et al., 2009, 2010); we are not aware of published contributions focusing

on the MoP framework. Romero et al. (2006) used a kernel estimator to represent the data distribution, and thereafter fitted an MTE to the kernel using regression. Langseth et al. (2009, 2010) attempted to find maximum likelihood parameters directly but since the maximum likelihood equations have no analytic solution in general, they instead proposed an iterative scheme utilizing Newton’s method and Lagrange multipliers. This resulted in better estimators (in terms of likelihood on the training-set as well as on a hold-out test-set), but at the cost of a steep increase in the computational complexity.

We present a new parameter estimation method, which aims at *approximating* the maximum likelihood parameters of an MoTBF network with known structure. We compare our results to those of Langseth et al. (2009, 2010), and find that although the new method finds parameters that are slightly poorer (in terms of likelihood), it is more than an order of magnitude faster than previous techniques.

The rest of the paper is organized as follows:

We start with an introduction to the MoTBF framework in Section 2. The simplified problem of learning univariate MoTBFs from data is considered in Section 3, and we discuss learning conditional distributions in Section 4. We report on some experiments in Section 5, and finally we conclude in Section 6.

## 2 The MoTBF model

The MoTBF framework is based on the abstract notion of real-valued *basis functions*  $\psi(\cdot)$ , which includes both polynomial and exponential functions as special cases. The first building-block of the framework is the marginal distribution: Let  $X$  be a continuous variable<sup>1</sup> with domain  $\Omega_X \subseteq \mathbb{R}$  and let  $\psi_i : \mathbb{R} \rightarrow \mathbb{R}$ , for  $i = 0, \dots, k$ , define a collection of real basis functions. We say that a function  $g_k : \Omega_X \mapsto \mathbb{R}_0^+$  is a *mixture of truncated basis functions* (MoTBF) potential of level  $k$  wrt.  $\Psi = \{\psi_0, \psi_1, \dots, \psi_k\}$  if  $g_k$  can be written as

$$g_k(x) = \sum_{i=0}^k a_i \psi_i(x), \quad (1)$$

where  $a_i$  are real numbers. The potential is a density if  $\int_{\Omega_X} g_k(x) dx = 1$ . Note that as opposed to the MTE and MoP definitions, a marginal MoTBF potential does not employ interval refinement to improve its expressive power.

Next, we turn to the MoTBF definition of conditional distributions, which mirrors the corresponding definition for MTEs. Thus, the influence a set of continuous parent variables  $\mathbf{Z}$  has on their child variable  $X$  is encoded only through the partitioning of the domain of  $\mathbf{Z}$ ,  $\Omega_{\mathbf{Z}}$ , into hyper-cubes, and not directly in the functional form of  $g_k^{(\ell)}(x|\mathbf{z})$  inside each hyper-cube  $\Omega_{\mathbf{Z}}^\ell$ . More precisely, for a partitioning  $\mathcal{P} = \{\Omega_{\mathbf{Z}}^1, \dots, \Omega_{\mathbf{Z}}^m\}$  of  $\Omega_{\mathbf{Z}}$ , the conditional MoTBF is

<sup>1</sup>In this paper we will often refer to MoTBF potentials defined only over continuous variables. In such cases, we understand, unless the contrary is specified, that all the claims about such potentials are extensible to those potentials also containing discrete variables in their domains, simply by having the claims hold for each configuration of the discrete variables.

defined for  $\mathbf{z} \in \Omega_{\mathbf{Z}}^j$ ,  $1 \leq j \leq m$ , as

$$g_k^{(j)}(x|\mathbf{z} \in \Omega_{\mathbf{Z}}^j) = \sum_{i=0}^k a_{i,j} \psi_i(x). \quad (2)$$

Finally, the joint MoTBF distribution over  $\mathbf{x} = (x_1, \dots, x_n)$  is found using the usual factorization,  $g_{\mathbf{k}}(\mathbf{x}) = \prod_{i=1}^n g_{k_i}(x_i|\text{pa}(x_i))$ , where the marginals and conditional distributions are defined using Equations (1) and (2), respectively.

Langseth et al. (2012) describe a “translation” procedure for efficiently finding an MoTBF approximation of any density function. The approximation procedure assumes that the basis functions  $\Psi$  are both *legal* and *orthonormal*: If  $\mathcal{Q}$  is the set of all linear combinations of the members of a set of basis functions  $\Psi = \{\psi_i(\cdot)\}_{i=0}^\infty$ , then  $\Psi$  is said to be a *legal* set of basis functions if the following conditions hold:

- $\psi_0$  is constant in its argument.
- If  $\phi_i \in \mathcal{Q}$  and  $\phi_j \in \mathcal{Q}$ , then  $(\phi_i \cdot \phi_j) \in \mathcal{Q}$ .
- For any pair of real numbers  $s$  and  $t$ , there exists a function  $\phi \in \mathcal{Q}$  such that  $\phi(s) \neq \phi(t)$ .

When considering orthonormal basis functions, we focus on the space  $L^2[a, b]$  of quadratically integrable real functions over the finite interval  $\Omega = [a, b]$ . For two functions  $\phi_i$  and  $\phi_j$  defined on  $\Omega$  we define the inner product as

$$\langle \phi_i, \phi_j \rangle = \int_{\Omega} \phi_i(x) \phi_j(x) dx,$$

and say that two functions are orthonormal if  $\langle \phi_i, \phi_j \rangle = \delta_{ij}$ , where  $\delta_{ij}$  is the Kronecker delta. A set of non-orthonormal basis functions can easily be orthonormalized using, for instance, the Gram-Schmidt procedure.

To set the scene, we let  $f(x)$  be the (target) density, and let  $g_k(x|\boldsymbol{\theta})$  be an MoTBF of order  $k$ . The key idea of Langseth et al. (2012) is to use *generalized Fourier series* to find “optimal” values for  $\boldsymbol{\theta} = (\theta_0, \dots, \theta_k)$ , that is, choosing  $\hat{\theta}_i = \langle f, \psi_i \rangle$ . It can easily be shown that while the generalized Fourier approximation up to degree  $k$  is guaranteed to minimize the  $L^2$  distance

$\int_x \left( f(x) - g_k(x|\hat{\boldsymbol{\theta}}) \right)^2 dx$ ,  $g_k$  is not always positive, and is thus not a density approximation. A convex optimization scheme (initialized with the generalized Fourier series coefficients) was therefore employed to obtain parameters that guarantee that  $g_k(x)$  is a density, and at the same time minimize an upper bound of the KL divergence  $D(f \| g_k)$  (Langseth et al., 2012). It was also shown that the approximation can be made arbitrarily tight, simply by increasing  $k$ .

### 3 Learning univariate distributions

While Langseth et al. (2012) defined their translation procedure as a means to create MoTBF approximations of *known* distributions, this paper will utilize the translation for *learning* hybrid BNs from data. The top-level algorithm is to (1) approximate the data using a kernel-density, and (2) approximate the kernel density with an MoTBF parameterization. We discuss each step below, and start by looking at univariate (marginal) distributions. We will move on to conditional distributions in Section 4.

Assume that  $f(x)$  is the (unknown) density, which generated the univariate sample  $\mathcal{D} = \{x_1, \dots, x_N\}$ . Next, let  $h_{\mathcal{D}}(\cdot|t_w)$  be a kernel density estimator based on the samples  $\mathcal{D}$  using kernel function  $t_w$  with bandwidth  $w$ . We define the kernel density estimator s.t.  $w$  approaches zero as  $N \rightarrow \infty$ . Now, the soundness of the approach rests upon the following proposition:

**Proposition 1.** *Let  $\tilde{\boldsymbol{\theta}}_N$  be chosen to minimize  $D(h_{\mathcal{D}}(x|t_w) \| g_k(x|\tilde{\boldsymbol{\theta}}_N))$ . Then  $\tilde{\boldsymbol{\theta}}_N$  converges to the maximum likelihood estimator of  $\boldsymbol{\theta}$  as  $N \rightarrow \infty$ .*

#### Sketch of proof:

First we note that since

$$\begin{aligned} D(h_{\mathcal{D}}(x|t_w) \| g_k(x|\boldsymbol{\theta})) &= \\ \int_x h_{\mathcal{D}}(x|t_w) \log \left( \frac{h_{\mathcal{D}}(x|t_w)}{g_k(x|\boldsymbol{\theta})} \right) dx, \end{aligned}$$

minimizing  $D(h_{\mathcal{D}}(x|t_w) \| g_k(x|\boldsymbol{\theta}))$  wrt.  $\boldsymbol{\theta}$  is equivalent to maximizing  $\mathbb{E}_{h_{\mathcal{D}}}[\log g_k(X|\boldsymbol{\theta})]$  wrt.  $\boldsymbol{\theta}$ ; the expectation is taken wrt.  $X$ , which is assumed to have density function  $h_{\mathcal{D}}(\cdot|t_w)$ . Next,

since the bandwidth of  $h_{\mathcal{D}}(\cdot|t_w)$  decreases to 0 as  $N \rightarrow \infty$ , we have that

$$h_{\mathcal{D}}(x|t_w) \rightarrow \frac{1}{N} \sum_{\ell=1}^N \delta(x - x_{\ell})$$

as  $N \rightarrow \infty$ , where  $\delta(\cdot)$  is Dirac's delta. Therefore,  $\mathbb{E}_{h_{\mathcal{D}}}[\log g_k(X|\boldsymbol{\theta})] \rightarrow \int_x \sum_{\ell} \frac{1}{N} \delta(x - x_{\ell}) \cdot \log g_k(x|\boldsymbol{\theta}) dx = \frac{1}{N} \sum_{\ell} \log g_k(x_{\ell}|\boldsymbol{\theta})$ . It follows that the parameters that minimize the KL divergence are asymptotically also those that maximize the likelihood.

The MoTBF density  $g_k(x|\boldsymbol{\theta})$  uses  $k+2$  parameters: The interval of support (2 values) and the  $k$  “free”  $\theta$ -values ( $\theta_0$  is fixed to make sure the function integrates to one; recall that an MoTBF density is defined without interval refinement). Thus, we can choose between models  $g_e$  and  $g_m$  using an (approximate) BIC-score: The (approximate) ML parameters are found, and each model is penalized according to complexity. In the experiments reported in Section 5 we have used a greedy approach starting from  $g_0$  and stopping as soon as an approximation  $g_k$  is better (in terms of BIC) than both  $g_{k+1}$  and  $g_{k+2}$ .<sup>2</sup> Our greedy procedure is exemplified in Figure 1, where an MoTBF is learned from 50 samples from a standard Gaussian distribution (shown as crosses on the  $x$ -axis). The kernel density approximation is drawn with a dashed line, and the MoTBF approximations from  $g_0$  up to  $g_{10}$  are shown;  $g_5$  is the best in terms of BIC-score, and is drawn with double line-width.

To fully specify the learning of univariate MoTBF distributions from data, we need to further analyze the use of kernel density approximations as an intermediate representation between the data and the learned MoTBF. The use of kernel estimators for learning MTEs was first proposed by Romero et al. (2006), and further analyzed by Langseth et al. (2010). The Epanechnikov kernel was found to offer the most consistent results in the case of MTE learning (Langseth et al., 2010), but we nevertheless use

<sup>2</sup>Computationally more demanding procedures can also be devised, e.g., to compare all subsets of basis functions from a fixed set  $\{\psi_0, \psi_1, \dots, \psi_k\}$ .

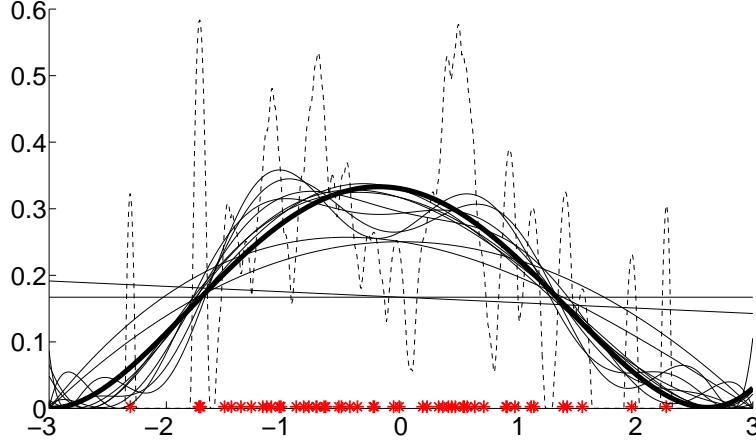


Figure 1: BIC-based learning: 50 samples from a standard Gaussian (crosses on the  $x$ -axis) are evaluated. The density estimator (thin dashed line) is the target of the MoTBF translations.  $g_0$  up to  $g_{10}$  are shown;  $g_5$  is the best in terms of BIC-score, and is drawn with double line-width.

the Gaussian kernel in our work to speed up the implementation. Previous attempts used Silverman’s rule of thumb when selecting the bandwidth,  $t_w^S \approx 1.06 \cdot \hat{\sigma} \cdot N^{-1/5}$ , where  $\hat{\sigma}$  is the empirical standard deviation of the dataset. By following that procedure, we get the results shown in Figure 2 (left-hand part of figure): a kernel density estimator is fitted to 50 samples from a standard Gaussian distribution, and the kernel density (drawn with the thin line) is then used as a starting point for the BIC-based MoTBF learning. The learned MoTBF representation is defined using 3 basis functions. Visually, the MoTBF approximation is quite poor (it does not resemble the standard Gaussian drawn with a dashed line), and we argue that the reason for the poor result is that using  $t_w^S$  is an unfortunate bandwidth choice, as it in principle leads us to smoothing the data *twice*: once when employing the kernel density, and once when the MoTBF is fitted to the kernel density. Rather, we want the kernel density to be a faithful representation of the data. To illustrate the effect, the right-hand part of Figure 2 shows the result of using the scaled bandwidth  $t_w^S/25$ . For this bandwidth, the BIC-score is optimized using 5 basis functions. The results are visually more appealing, and this is underlined when calculating the

log likelihood of a hold-out set, giving  $-1541.02$  and  $-1464.86$  for the two bandwidths, respectively. We have investigated this further by examining a range of different datasets, both small and large, as defined by Langseth et al. (2010). For each dataset, we have learned an MoTBF representation using the BIC score for model selection and with a set of bandwidths defined by  $t_w \leftarrow t_w^S/\alpha$ , where  $\alpha \in \{1, 2, 5, 10, 25, 50\}$  is the bandwidth scale. Table 1 lists the results of the experiment in terms of the number of basis functions that are selected as well as the obtained log likelihood on a hold-out dataset. The results appear to be robust across the data sets as long as the bandwidth is “sufficiently small”. We have therefore used a fixed value of  $t_w^S/10$  in the following, unless stated otherwise.

#### 4 Learning conditional distributions

Recall that for a conditional MoTBF  $f(x|\mathbf{z})$ , the variables  $\mathbf{Z}$  influence  $X$  only through the partitioning of  $\Omega_{\mathbf{Z}}$ , see Equation (2). Learning a conditional MoTBF therefore amounts to finding

- a partitioning of  $\Omega_{\mathbf{Z}}$ , and
- a (univariate) MoTBF for each hyper-cube in the partitioning.

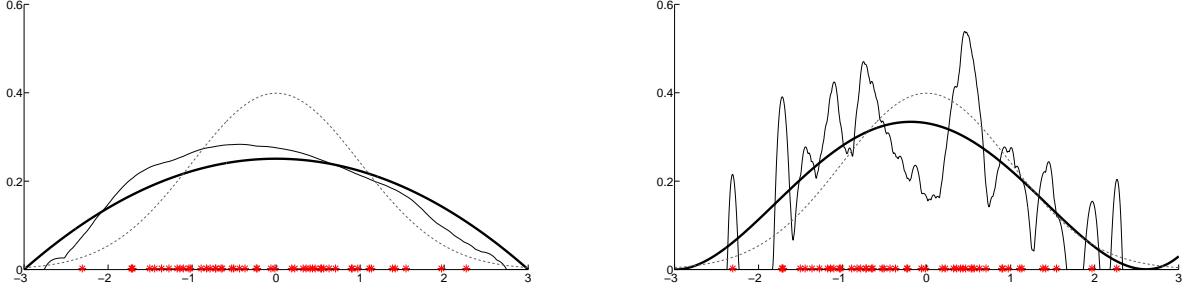


Figure 2: 50 samples from a Gaussian distribution are learned using a kernel density. The kernel density is shown in a thin line, where the bandwidth is the Silverman's rule of thumb (left figure) and one twenty-fifth of Silverman's rule of thumb (right). The learned MoTBF representations are defined using 3 and 5 basis functions in the left and right hand plots, respectively, and drawn in a thick line. Visually, the right-hand figure gives a better fit (compare to the standard Gaussian density drawn with the dashed line), and this is also the case when evaluated using log likelihood of a hold-out set ( $-1541.020$  and  $-1465.585$ , respectively).

The algorithm for learning conditionals MoTBFs proceeds by iterating over the two steps above.

#### 4.1 Finding an MoTBF for a fixed partitioning

For a given hyper-cube  $\Omega_Z^l \in \mathcal{P}$  we start by approximating the conditional empirical distribution with a conditional kernel density estimate. For ease of exposition, consider a variable  $Y$  with parent  $X$  for which we have a data sample  $\mathcal{D} = \{\mathbf{d}_1, \dots, \mathbf{d}_N\}$ , where  $\mathbf{d}_i = (x_i, y_i)$ . We now define the conditional kernel density estimate for  $Y$  given  $X$  as

$$h_{\mathcal{D}}(y|x, t_{w_y}, t_{w_x}) = \frac{\sum_{i=1}^N h_{y_i}(y|t_{w_y})h_{x_i}(x|t_{w_x})}{\sum_{i=1}^N h_{x_i}(x|t_{w_x})},$$

where  $h_{x_i}(x|t_{w_x})$  is a kernel density estimator based on  $x_i$  only and with bandwidth  $t_{w_x}$ ;  $h_{y_i}(y|t_{w_y})$  is defined similarly. Given a conditional kernel density estimator  $h_{\mathcal{D}}(y|x, t_{w_y}, t_{w_x})$  and a partitioning  $\mathcal{P}$  of  $\Omega_X$ , we approximate  $h_{\mathcal{D}}(y|x, t_{w_y}, t_{w_x})$  with an MoTBF potential  $f(y|x)$  (see Equation (2)) by following the procedure of Langseth et al. (2012). Thus, for all  $\Omega_X^l \in \mathcal{P}$ , we seek

$$f(y|x \in \Omega_X^l) \sim h_{\mathcal{D}}(y|x \in \Omega_X^l, t_{w_y}, t_{w_x}) = \int_x h_{\mathcal{D}}(y|x, t_{w_y}, t_{w_x})h_{\mathcal{D}}(x|x \in \Omega_X^l, t_{w_x})dx,$$

where the integral can be approximated by  $\sum_{i=1}^n h_{\mathcal{D}}(y|x_i, t_{w_y}, t_{w_x})h_{\mathcal{D}}(x_i|x_i \in \Omega_X^l, t_{w_x})$  using data samples  $x_1, \dots, x_n$  from  $\mathcal{D}$  belonging to  $\Omega_X^l$ . That is, for a fixed partitioning of  $\Omega_X$  learning a conditional MoTBF potential reduces to estimating a univariate MoTBF potential (as described in Section 3) for each partition  $\Omega_X^l \in \mathcal{P}$ .

#### 4.2 Finding a partitioning of the conditioning variables

In order to find a partitioning of  $\Omega_Z$  we employ a myopic strategy, where we in each step consider a bisection of an existing partition along each  $Z \in \mathbf{Z}$ . That is, for each partition  $\Omega_Z^l \in \mathcal{P}$  the algorithm evaluates the potential gain of splitting the partition along  $Z \in \mathbf{Z}$ . After scoring the candidate partitions the algorithm selects the highest scoring partition  $\Omega_Z^{BS}$  and splitting variable  $Z_{BS}$ , and learns MoTBF representations of the two induced sub-partitions  $\Omega_Z^{BS,1}$  and  $\Omega_Z^{BS,2}$ . To guide the selection of a candidate partition  $\Omega_Z'$  we consider the potential improvement in BIC score resulting from splitting that partition:

$$\text{BIC-Gain}(\Omega_Z', Z) = \text{BIC}(f', \mathcal{D}) - \text{BIC}(f, \mathcal{D}),$$

where  $f'$  is the conditional MoTBF potential defined over the partitioning  $\{\mathcal{P} \setminus \Omega_Z'\} \cup$

| Distribution  | Scaler $\alpha$ | $N = 50$ |          | $N = 1000$ |          |
|---------------|-----------------|----------|----------|------------|----------|
|               |                 | #BF      | Loglik   | #BF        | Loglik   |
| MTE           | 1               | 4        | -2430.82 | 5          | -2349.78 |
|               | 2               | 6        | -2349.76 | 9          | -2315.24 |
|               | 5               | 6        | -2349.76 | 12         | -2308.68 |
|               | 10              | 6        | -2349.76 | 12         | -2308.68 |
|               | 25              | 6        | -2349.76 | 9          | -2315.24 |
|               | 50              | 6        | -2349.76 | 9          | -2315.24 |
| Beta(.5,.5)   | 1               | 5        | 180.06   | 7          | 228.97   |
|               | 2               | 5        | 180.06   | 7          | 228.97   |
|               | 5               | 3        | 80.92    | 7          | 228.97   |
|               | 10              | 3        | 80.92    | 7          | 228.97   |
|               | 25              | 5        | 180.06   | 5          | 206.18   |
|               | 50              | 1        | 0.00     | 3          | 151.94   |
| $\chi_8^2$    | 1               | 2        | -2887.23 | 4          | -2803.42 |
|               | 2               | 4        | -2801.10 | 7          | -2736.99 |
|               | 5               | 5        | -2747.73 | 10         | -2711.65 |
|               | 10              | 5        | -2747.73 | 10         | -2711.65 |
|               | 25              | 5        | -2747.73 | 10         | -2711.65 |
|               | 50              | 5        | -2747.73 | 10         | -2711.65 |
| Gauss(0,1)    | 1               | 3        | -1541.02 | 6          | -1428.35 |
|               | 2               | 5        | -1464.86 | 7          | -1420.33 |
|               | 5               | 5        | -1464.86 | 7          | -1420.33 |
|               | 10              | 5        | -1464.86 | 7          | -1420.33 |
|               | 25              | 5        | -1464.86 | 7          | -1420.33 |
|               | 50              | 3        | -1541.02 | 5          | -1434.28 |
| Log-Norm(0,1) | 1               | 8        | -1434.51 | 7          | -1393.15 |
|               | 2               | 8        | -1434.51 | 8          | -1390.39 |
|               | 5               | 8        | -1434.51 | 8          | -1378.77 |
|               | 10              | 5        | -1523.29 | 9          | -1378.77 |
|               | 25              | 5        | -1523.29 | 8          | -1390.39 |
|               | 50              | 5        | -1523.29 | 8          | -1390.39 |

Table 1: The effect of the chosen bandwidth wrt. log likelihood of a test-set. In general, we see that results for “large” bandwidths ( $\alpha = 1$ ) are poor due to “double smoothing”. Additionally, results using large scalers ( $\alpha = 50$ ) are also sometimes unsatisfactory; typically due to numerical instabilities in the solution method due to the peakedness of the kernel approximation, which in turn leads to numerically unstable calculations.

$\{\Omega_{\mathbf{Z}}^{Z,1}, \Omega_{\mathbf{Z}}^{Z,2}\}$ . In principle, when scoring the model  $f'$  one would need to find the basis functions (and the corresponding parameters) maximizing this score. This will, however, be computationally difficult, and instead we lower-bound the improvement in BIC score by using the same set of basis functions as was used for the parent partition  $\Omega'_{\mathbf{Z}}$ . It should be noted that for the calculation of the improvement in BIC score, we only need to consider the parts of the score relating to the partition  $\Omega'_{\mathbf{Z}}$ , since the contributions from the partitions for which  $f$  and  $f'$  agree cancel out; this property also sup-

### Algorithm 1 Learning conditional MoTBFs.

```

1: $\mathcal{P} \leftarrow \{\Omega_{\mathbf{Z}}\}$
2: repeat
3: $(\Omega_{\mathbf{Z}}^{\text{BS}}, Z_{\text{BS}}) \leftarrow$
 $\arg \max_{\Omega'_{\mathbf{Z}} \in \mathcal{P}, Z \in \mathbf{Z}} \text{BIC-Gain}(\Omega'_{\mathbf{Z}}, Z)$
4: if $\text{BIC-Gain}(\Omega_{\mathbf{Z}}^{\text{BS}}, Z_{\text{BS}}) > 0$ then
5: Learn MoTBF potentials for
 $\Omega_{\mathbf{Z}}^{\text{BS},1}$ and $\Omega_{\mathbf{Z}}^{\text{BS},2}$.
6: $\mathcal{P} \leftarrow \{\mathcal{P} \setminus \Omega_{\mathbf{Z}}^{\text{BS}}\} \cup \{\Omega_{\mathbf{Z}}^{\text{BS},1}, \Omega_{\mathbf{Z}}^{\text{BS},2}\}$.
7: else
8: terminate.
9: end if
10: until false

```

ports an efficient caching scheme for BIC-Gain. The overall procedure for learning conditional MoTBFs is summarized in Algorithm 1.

## 5 Experiments

In this section we will report on two small experimental studies undertaken to compare the merits of the proposed method to its most immediate competitors. Firstly, we will compare the new method of learning marginal MoTBF densities to the results obtained by Romero et al. (2006) and Langseth et al. (2010), as reported by Langseth et al. (2010). Datasets, each containing 1000 training examples, generated from five different distributions were used. Table 2 reports the log likelihood of each dataset using the obtained estimator of each of the techniques. We have used the polynomials as basis functions, meaning that  $\psi_\ell$  in Equations (1) and (2) is the (scaled and stretched) Legendre polynomial of order  $\ell$ . The number of basis functions was chosen so that the number of free parameters corresponds to the number of parameters used by Langseth et al. (2010); recall that the MoTBF distribution  $g_k$  on  $\Omega_{\mathbf{Z}}$  is specified using  $k + 2$  parameters. Note that where the methods by Romero et al. (2006) and Langseth et al. (2010) divide the support of the distribution into sub-intervals and fit one model per interval, the current approach does not use interval refinement. This may harm the fit of the MoTBF distributions, when the gold-standard distribu-

| Dataset    | #par | Romero et al. (2006) | Langseth et al. (2010) | Current approach |
|------------|------|----------------------|------------------------|------------------|
| MTE        | 12   | -2556.68             | -2263.13               | -2272.71         |
| Beta       | 12   | 39.42                | 160.69                 | 159.74           |
| $\chi_8^2$ | 24   | -2766.86             | -2685.76               | -2710.35         |
| Gaussian   | 12   | -1565.28             | -1420.34               | -1387.73         |
| Log-Normal | 24   | -1636.99             | -1398.30               | -1373.73         |

Table 2: The obtained log likelihood of the training data when learning from 1000 samples from different distributions.

tion is not continuous. In general, though, the results of the new method seem to be better than those by Romero et al. (2006), and comparable to those by Langseth et al. (2010).

The speedup from the direct maximum likelihood approach (Langseth et al., 2010) to our approach is above a factor 10. The main contribution to the speed increase is that while the previous technique was based on an iterative scheme, where each potential solution (living in a very complicated likelihood-landscape) needed to be evaluated using a computationally expensive procedure, the current approach casts the learning problem as a convex optimization problem, with much cheaper evaluations.

Next, we compare the predictive performance of the method in Langseth et al. (2010) to ours. We used the same training data as reported in Table 2, but this time used the (approximate) BIC score for model selection. The chosen models were examined by calculating the log likelihood of a separate dataset of 1000 cases.

| Dataset    | Langseth et al. (2010) | Current approach |
|------------|------------------------|------------------|
| MTE        | -2285.25               | -2308.68         |
| Beta       | 249.45                 | 228.97           |
| $\chi_8^2$ | -2702.67               | -2711.65         |
| Gaussian   | -1430.88               | -1420.33         |
| Log-Normal | -1358.38               | -1378.77         |

Table 3: Test-set log likelihood of estimators after learning models using the BIC score.

The results in Table 3 indicate that the method by Langseth et al. (2010) is slightly better than our procedure, but the speedup of the current approach is more than a factor 15. The main source of the extra speed-increase is that the relatively costly initialization of the MoTBF technique (finding and representing the orthonormal

basis functions) needs not be performed each time a new candidate model is evaluated.

Finally, we exemplify the learning of conditional distributions by generating data from a model where  $X$  is standard Gaussian, and  $Y|X=x \sim \mathcal{N}(x/2, 1)$ . Datasets containing 50, 500, 2500 and 5000 cases were generated, and given to Algorithm 1. The resulting conditional distributions are shown in Figure 3, with the parent on the  $x$ -axis and the child on the  $y$ -axis. For the smallest dataset of 50 cases, the BIC-score only gave support for a single split-point, inserted at the midpoint of the support for  $X$ . As the size of the training-sets increases, the BIC score selects more and more refined models, allowing itself to use more parameters to represent the correlation between  $X$  and  $Y$  as it is more and more clearly manifested in the training data. Notice that the algorithm uses more effort on refining the model where the bulk of the data is found (i.e., around  $x \approx 0$ ).

## 6 Conclusions

In this paper we have examined a new technique for (approximately) learning maximum likelihood parameters of a hybrid Bayesian network from data. The main idea is to find a kernel density estimate of the data and utilize an effective “translation” procedure designed for approximating any marginal or conditional distribution function (in this case a kernel density) by an MoTBF distribution. Although the method was found to be slightly worse than state-of-the-art techniques in terms of the log-likelihood score, the speed-up over previous methods is substantial, and we are currently investigating how the method scales to larger domains.

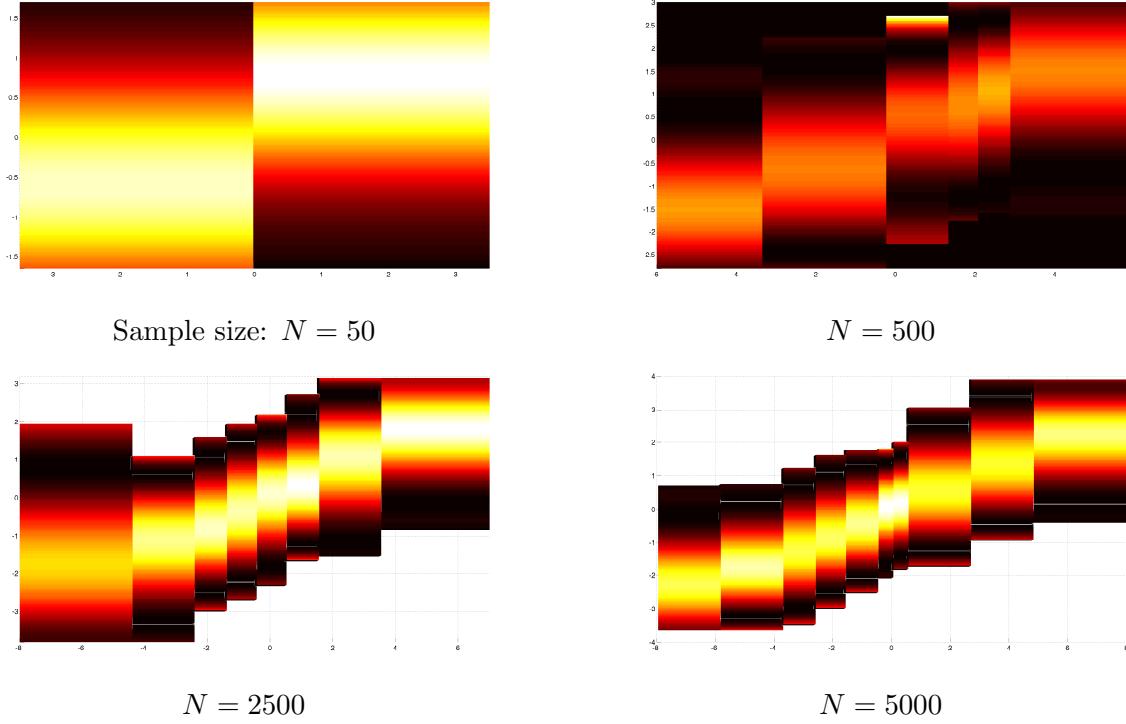


Figure 3: Learning a conditional linear Gaussian distribution using BIC score. Note how finer model granularity is selected as the size of training-set grows, and how the discretization effort is kept to the area with the bulk of the data.

## Acknowledgments

This work has been supported by a Senior Grant in the frame of the CALL UCM-EEA-ABEL-02-2009 of the Abel Extraordinary Chair (NILS Project), and by the Spanish Ministry of Science and Innovation, through projects TIN2010-20900-C04-02,03 (entitled Data mining with PGMs: New algorithms and applications) and by ERDF (FEDER) funds.

## References

- Langseth, H., Nielsen, T., Rumí, R., and Salmerón, A. (2009). Maximum likelihood learning of conditional MTE distributions. *ECSQARU 2009. Lecture Notes in Computer Science*, 5590:240–251.
- Langseth, H., Nielsen, T., Rumí, R., and Salmerón, A. (2010). Parameter estimation and model selection for mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 51:485–498.
- Langseth, H., Nielsen, T., Rumí, R., and Salmerón, A. (2012). Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53:212–227.
- Moral, S., Rumí, R., and Salmerón, A. (2001). Mixtures of truncated exponentials in hybrid Bayesian networks. In *ECSQARU'01. Lecture Notes in Artificial Intelligence*, volume 2143, pages 135–143.
- Romero, V., Rumí, R., and Salmerón, A. (2006). Learning hybrid Bayesian networks using mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 42:54–68.
- Shafer, G. R. and Shenoy, P. P. (1990). Probability propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–352.
- Shenoy, P. and West, J. (2011). Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52:641–657.

# Inference in hybrid Bayesian networks with Mixtures of Truncated Basis Functions

Helge Langseth

Department of Computer and Information Science  
The Norwegian University of Science and Technology  
Trondheim (Norway)  
[helgel@idi.ntnu.no](mailto:helgel@idi.ntnu.no)

Thomas Dyhre Nielsen

Department of Computer Science  
Aalborg University  
Aalborg (Denmark)  
[tdn@cs.aau.dk](mailto:tdn@cs.aau.dk)

Rafael Rumí

Department of Statistics and Applied Mathematics  
University of Almería, Almería (Spain)  
[rrumi@ual.es](mailto:rrumi@ual.es)

Antonio Salmerón

Department of Statistics and Applied Mathematics  
University of Almería, Almería (Spain)  
[antonio.salmeron@ual.es](mailto:antonio.salmeron@ual.es)

## Abstract

In this paper we study the problem of exact inference in hybrid Bayesian networks using mixtures of truncated basis functions (MoTBFs). We propose a structure for handling probability potentials called Sum-Product factorized potentials, and show how these potentials facilitate efficient inference based on *i*) properties of the MoTBFs and *ii*) ideas similar to the ones underlying Lazy propagation (postponing operations and keeping factorized representations of the potentials). We report on preliminary experiments demonstrating the efficiency of the proposed method in comparison with existing algorithms.

## 1 Introduction

Inference in hybrid Bayesian networks has received considerable attention over the last decade. In order to perform exact propagation in hybrid domains, the main challenge is to find a representation of the joint distribution that supports an efficient implementation of the usual inference operators: marginalization, and combination.

If the joint distribution belongs to e.g. the class of *conditional Gaussian* distributions (Lauritzen, 1992; Lauritzen and Jensen, 2001; Olesen, 1993), then inference can be performed exactly. However for this model class the continuous variables are assumed to follow a linear Gaussian distribution and the discrete variables are not allowed to have continuous parents. The mixture of truncated exponentials (MTE) model (Moral et al., 2001) does not impose such restrictions, but instead allows continuous and discrete variables to be treated in a uniform fashion. Furthermore, the MTE model class supports both exact (Cobb et al., 2004)

and approximate inference methods (Rumí and Salmerón, 2007). Recently, the mixtures of polynomials (MOPs) model has been proposed as an alternative to the MTE model (Shenoy and West, 2011); the MOP model shares the advantages of MTEs, but it also provides a more flexible way of handling deterministic relationships among variables (Shenoy, 2011).

A more general approach for representing hybrid Bayesian networks has been introduced by Langseth et al. (2012) in the form of the mixtures of truncated basis functions (MoTBFs) model. MoTBFs are based on general real-valued basis functions that includes exponential and polynomial functions as special cases. Langseth et al. (2012) also show that efficient algorithms can be devised for approximating arbitrary probability density functions using MoTBFs.

In this paper we explore the problem of exact inference in hybrid Bayesian networks, where the potentials are specified using MoTBFs. We propose an algorithm that exploits properties of

the basis functions and makes use of the ideas behind *Lazy propagation* in discrete networks (Madsen and Jensen, 1999; Madsen, 2010), i.e., postponing operations as long as possible and keeping factorized representations of the probability potentials. Preliminary experimental results show that the proposed algorithm provides significant improvements in efficiency compared to existing inference procedures for MTEs.

## 2 Preliminaries

The MoTBF potential was proposed by Langseth et al. (2012) as an alternative to the MTE and the MOP models, for which the exponential and polynomial functions are replaced by the more abstract notion of a basis function.<sup>1</sup>

**Definition 1.** Let  $\mathbf{X}$  be a mixed  $n$ -dimensional random vector. Let  $\mathbf{Y} = (Y_1, \dots, Y_d)$  and  $\mathbf{Z} = (Z_1, \dots, Z_c)$  be the discrete and continuous parts of  $\mathbf{X}$ , respectively, with  $c + d = n$ . Let  $\Psi = \{\psi_i(\cdot)\}_{i=0}^{\infty}$  with  $\psi_i : \mathbb{R} \rightarrow \mathbb{R}$  define a collection of real basis functions. A function  $\hat{f} : \Omega_{\mathbf{X}} \rightarrow \mathbb{R}_0^+$  is a mixture of truncated basis functions (MoTBF) potential of level  $k$  wrt.  $\Psi$  if one of the following two conditions holds:

1.  $\hat{f}$  can be written as

$$\hat{f}(\mathbf{x}) = \hat{f}(\mathbf{y}, \mathbf{z}) = \prod_{j=1}^c \sum_{i=0}^k a_{i,\mathbf{y}}^{(j)} \psi_i(z_j), \quad (1)$$

where  $a_{i,\mathbf{y}}^{(j)}$  are real numbers.

2. There is a partitioning  $\Omega_{\mathbf{X}}^1, \dots, \Omega_{\mathbf{X}}^m$  of  $\Omega_{\mathbf{X}}$  for which the domain of the continuous variables,  $\Omega_{\mathbf{Z}}$ , is divided into hyper-cubes such that  $\hat{f}$  is defined as

$$\hat{f}(\mathbf{x}) = f^{(l)}(\mathbf{x}) \text{ if } \mathbf{x} \in \Omega_{\mathbf{X}}^l, \quad (2)$$

where each  $f^{(l)}$ ,  $l = 1, \dots, m$ , can be written in the form of Equation (1).

An MoTBF potential is a density if  $\sum_{\mathbf{y} \in \Omega_{\mathbf{Y}}} \int_{\Omega_{\mathbf{Z}}} \hat{f}(\mathbf{y}, \mathbf{z}) d\mathbf{z} = 1$ .

<sup>1</sup>We give a definition which is slightly altered compared to the original definition by Langseth et al. (2012). This subtle difference is introduced to simplify the following deductions and will not have practical implications.

An MoTBF  $f(\mathbf{z}_1, \mathbf{z}_2, \mathbf{y})$  defined over the continuous variables  $\mathbf{Z}_1$  and  $\mathbf{Z}_2$  and the discrete variables  $\mathbf{Y}$  is a *conditional MoTBF* for  $\mathbf{Z}_1$  if  $\int_{\mathbf{z}_1} f(\mathbf{z}_1, \mathbf{z}_2, \mathbf{y}) d\mathbf{z}_1 = 1$  for all  $\mathbf{z}_2 \in \Omega_{\mathbf{Z}_2}$  and  $\mathbf{y} \in \Omega_{\mathbf{Y}}$ .<sup>2</sup> Following Langseth et al. (2012) we assume that the conditioning variables only affect the density through the hyper-cubes over which the density is defined. Thus, for variables  $\mathbf{Z}_1$  and  $\mathbf{Z}_2$  with  $\Omega_{\mathbf{Z}_2}$  partitioned into  $\Omega_{\mathbf{Z}_2}^1, \dots, \Omega_{\mathbf{Z}_2}^m$  we define the conditional MoTBF density  $f(\mathbf{z}_1 | \mathbf{z}_2, \mathbf{y})$  as:

$$f(\mathbf{z}_1 | \mathbf{z}_2, \mathbf{y}) = \prod_{j=1}^c \sum_{i=0}^k a_{i,\mathbf{y},l}^{(j)} \psi_i(\mathbf{z}_1^{(j)}), \quad (3)$$

for  $\mathbf{z}_2 \in \Omega_{\mathbf{Z}_2}^l$ . Consequently, given a partitioning of the conditioning variables  $\mathbf{z}_2$ , finding a conditional MoTBF  $f(z_1 | \mathbf{z}_2)$  for a single variable  $z_1$  reduces to specifying a collection of univariate MoTBFs. By extension, specifying the distributions of a hybrid Bayesian network therefore involves univariate MoTBF potentials only, and the form of the MoTBF potentials simplifies to

$$\hat{f}(z_1 | \mathbf{z}_2, \mathbf{y}) = \sum_{i=0}^k a_{i,\mathbf{y},l}^{(1)} \psi_i(z_1),$$

for  $\mathbf{z}_2 \in \Omega_{\mathbf{Z}_2}^l$ .

The approximation procedure described in (Langseth et al., 2012) also assumes that the basis functions  $\Psi$  are both *legal* and *orthonormal*: If  $\mathcal{Q}$  is the set of all linear combinations of the members of a set of basis functions  $\Psi = \{\psi_i(\cdot)\}_{i=0}^{\infty}$ , then  $\Psi$  is said to be a *legal* set of basis functions if the following conditions hold:

- $\psi_0$  is constant in its argument.
- If  $f \in \mathcal{Q}$  and  $g \in \mathcal{Q}$ , then  $(f \cdot g) \in \mathcal{Q}$ .
- For any pair of real numbers  $s$  and  $t$ , there exists a function  $f \in \mathcal{Q}$  such that  $f(s) \neq f(t)$ .

Clearly, the sets of basis functions  $\{x^i\}_{i=0}^{\infty}$  and  $\{\exp(-i \cdot x), \exp(i \cdot x)\}_{i=0}^{\infty}$  are legal and correspond to examples of bases for the MOP and MTE frameworks, respectively.

<sup>2</sup>For ease of presentation we disregard possible partitionings of  $\Omega_{\mathbf{Z}_1}$ .

When considering orthonormal basis functions, we focus on the space  $L^2[a, b]$  of quadratically integrable real functions over the finite interval  $[a, b]$ . For two functions  $f(x)$  and  $g(x)$  defined on  $[a, b]$  we define the inner product as

$$\langle f, g \rangle = \int_a^b f(x)g(x)dx,$$

and say that two functions are orthonormal if and only if  $\langle f, g \rangle = 0$  and  $\langle f, f \rangle = \langle g, g \rangle = 1$ . A set of non-orthonormal basis functions can easily be orthonormalized using, for instance, the Gram-Schmidt procedure.

In this paper we will often refer to MoTBF potentials defined only over continuous variables. In such cases, we understand, unless specified otherwise, that all the claims about such potentials are extensible to those potentials also containing discrete variables in their domains, simply by having the claims hold for each configuration of the discrete variables. Furthermore, in the remainder of the paper we assume a fixed set of  $m$  basis functions for each continuous variable in the network, i.e.,  $\Psi = \{\psi_0, \psi_1, \dots, \psi_{m-1}\}$ .<sup>3</sup>

### 3 Operations over MoTBFs

There are three operations used by exact inference algorithms: restriction, combination, and marginalization. The first operation is trivial and is basically used to incorporate evidence prior to the inference process, while the others are used throughout the inference process.

**Definition 2** (Combination). *Let  $f_1(\mathbf{y}_1, \mathbf{z}_1)$  and  $f_2(\mathbf{y}_2, \mathbf{z}_2)$  be MoTBF potentials defined over the partitions  $\mathcal{P}_1 = \{\Omega_{\mathbf{Z}_1}^1, \dots, \Omega_{\mathbf{Z}_1}^{k_1}\}$  and  $\mathcal{P}_2 = \{\Omega_{\mathbf{Z}_2}^1, \dots, \Omega_{\mathbf{Z}_2}^{k_2}\}$  of  $\Omega_{\mathbf{Z}_1}$  and  $\Omega_{\mathbf{Z}_2}$ , respectively:*

$$f_h(\mathbf{y}_h, \mathbf{z}_h) = \prod_{j=1}^{c_h} \sum_{i=0}^{m-1} a_{i, \mathbf{y}, l_h}^{(j)} \psi_i(z_h^{(j)}),$$

for  $\mathbf{z}_h \in \Omega_{\mathbf{Z}_h}^{l_h}$  ( $\Omega_{\mathbf{Z}_h}^{l_h} \in \mathcal{P}_h$ ) and  $h = 1, 2$ . The combination of  $f_1$  and  $f_2$  is a potential  $f(\mathbf{y}, \mathbf{z})$

<sup>3</sup>Note that any potential defined over a subset  $\Psi' \subseteq \Psi$  can be represented by setting the coefficients of the potentials in  $\Psi \setminus \Psi'$  to 0.

over  $\mathbf{Y} = \mathbf{Y}_1 \cup \mathbf{Y}_2$  and  $\mathbf{Z} = \mathbf{Z}_1 \cup \mathbf{Z}_2$ :

$$f(\mathbf{y}, \mathbf{z}) = \prod_{j=1}^{c_1} \prod_{r=1}^{c_2} \left( \sum_{i=0}^{m-1} a_{i, \mathbf{y}_1, l_1}^{(j)} \psi_i(\mathbf{z}_1^{(j)}) \right) \left( \sum_{i=0}^{m-1} a_{i, \mathbf{y}_2, l_2}^{(r)} \psi_i(\mathbf{z}_2^{(r)}) \right) \quad (4)$$

for all  $\mathbf{z} \in \Omega_{\mathbf{Z}_1}^{l_1} \times \Omega_{\mathbf{Z}_2}^{l_2}$  and  $\mathbf{y} \in \Omega_{\mathbf{Y}_1}^{l_1} \times \Omega_{\mathbf{Y}_2}^{l_2}$ .

Observe that each factor in Equation (4) is an MoTBF. If the products in Equation (4) are not expanded any further, we call the operation *lazy combination*, pointing out the fact that no actual product is carried out; instead the factors in the original potentials are concatenated in a single list.

**Definition 3** (Factorized potential). *A potential defined over hyper-cubes, and where in each hyper-cube the potential is defined as a list of factors of the form given in Equation (4) is called a factorized potential.*

Generalizing the notion of lazy combination to factorized potentials follows immediately.

**Definition 4** (Marginalization of factorized potentials). *Let  $f_{\mathbf{Z}}$  be a factorized potential defined for variables  $\mathbf{Z} = \{Z_1, \dots, Z_c\}$  over hyper-cubes  $\bigcup_{h=1}^k \Omega_{\mathbf{Z}}^h$ . The result of marginalizing out a variable  $Z_j$  from  $f_{\mathbf{Z}}$  is a new potential defined on  $\mathbf{Z} \setminus \{Z_j\}$  over the hyper-cubes  $\bigcup_{h=1}^{k'} \Omega_{\mathbf{Z} \setminus \{Z_j\}}^h$ , where for each new hyper-cube  $\Omega_{\mathbf{Z} \setminus \{Z_j\}}^h$ ,  $h = 1, \dots, k'$  (obtained by projecting  $\Omega_{\mathbf{Z}}^h$  onto  $\mathbf{Z} \setminus \{Z_j\}$ ) the marginalized potential is defined as*

$$f_{\mathbf{Z} \setminus \{Z_j\}}(z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_c) = \sum_{l=1}^r \int_{\Omega_{Z_j}} f^{(h,l)}(z_1, \dots, z_c) dz_j,$$

where  $\Omega_{Z_j}^1 \cup \dots \cup \Omega_{Z_j}^r$  is the partition of the domain of  $Z_j$  in  $f_{\mathbf{Z}}$  and  $f^{(h,l)}$  denotes the value of potential  $f_{\mathbf{Z}}$  in hyper-cube  $\Omega_{\mathbf{Z} \setminus \{Z_j\}}^h \times \Omega_{Z_j}^l$ .

**Proposition 1.** *Let  $f_{\mathbf{Z}}$  be a factorized potential under the same conditions as in Definition 4. Then, for each hyper-cube  $\Omega_{\mathbf{Z} \setminus \{Z_j\}}^h$ ,*

$h = 1, \dots, k'$ ,

$$\begin{aligned} f_{\mathbf{Z} \setminus \{Z_j\}}(z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_c) \\ = \sum_{l=1}^r \left( \prod_{i \neq j} \sum_{s=0}^{m-1} a_{s,\cdot,(h,l)}^{(i)} \psi_s(z_i) \right). \end{aligned}$$

*Proof.* By expanding the integral in Def. 4 we get

$$\begin{aligned} & \int f^{(h,l)}(z_1, \dots, z_c) dz_j \\ &= \int \prod_{i=1}^c \left( \sum_{s=0}^{m-1} a_{s,\cdot,(h,l)}^{(i)} \psi_s(z_i) \right) dz_j \\ &= \prod_{i \neq j} \left( \sum_{s=0}^{m-1} a_{s,\cdot,(h,l)}^{(i)} \psi_s(z_i) \right) \int \sum_{s=0}^{m-1} a_{s,\cdot,(h,l)}^{(j)} \psi_s(z_j) dz_j \\ &= \prod_{i \neq j} \left( \sum_{s=0}^{m-1} a_{s,\cdot,(h,l)}^{(i)} \psi_s(z_i) \right) \left( \sum_{s=0}^{m-1} a_{s,\cdot,(h,l)}^{(j)} \int \psi_s(z_j) dz_j \right). \end{aligned}$$

Since the basis functions  $\psi_s$ ,  $s = 0, \dots, m-1$  are orthonormal, it holds that  $\int_{\Omega_{Z_j}} \psi_{s_1}(z_j) \psi_{s_2}(z_j) dz_j = 0$  for any  $s_1 \neq s_2 \in \{0, \dots, m-1\}$ . Moreover, taking into account that  $\psi_0(z_j)$  is a constant, it follows that  $\int_{\Omega_{Z_j}} \psi_s(z_j) dz_j = 0$  for any  $s > 0$ . Furthermore, if  $s = 0$  we have that

$$a_{0,\cdot,(h,l)}^{(j)} \int_{\Omega_{Z_j}} \psi_0(z_j) dz_j = 1.$$

Hence,

$$\begin{aligned} & \int_{\Omega_{Z_j}} f^{(h,l)}(z_1, \dots, z_c) dz_j \\ &= \prod_{i \neq j} \left( \sum_{s=0}^{m-1} a_{s,\cdot,(h,l)}^{(i)} \psi_s(z_i) \right). \end{aligned}$$

□

From Prop. 1 we see that the result of marginalizing out a variable from a factorized potential is not necessarily a factorized potential, but rather a sum of factorized potentials. We therefore need to extend the concept of factorized potentials in order to allow factorized representations with respect to sums and products.

**Definition 5** (SP factorized potential). Let  $f_{\mathbf{Z}}$  be a potential defined for variables  $\mathbf{Z} = \{Z_1, \dots, Z_c\}$  over hyper-cubes  $\bigcup_{i=1}^k \Omega_{\mathbf{Z}}^i$ . We say that  $f_{\mathbf{Z}}$  is a Sum-Product (SP) factorized potential if it can be written as

$$f_{\mathbf{Z}}(\mathbf{z}) = \sum_{j=1}^t f_{\mathbf{Z}}^{(j)}(\mathbf{z}), \quad (5)$$

where  $t > 0$  and  $f^{(j)}$ ,  $j = 1, \dots, t$ , are factorized potentials according to Definition 3.

**Corollary 1.** The result of marginalizing out a variable from a factorized potential is an SP factorized potential.

*Proof.* It follows directly from the proof of Proposition 1. □

**Definition 6** (Combination of SP factorized potentials). Let

$$f_{\mathbf{X}_i}(\mathbf{x}_i) = \sum_{l=1}^{r_i} f_{\mathbf{X}_i}^{(l)}(\mathbf{x}_i) \quad \text{for } i = 1, 2$$

be two SP factorized potentials over variables  $\mathbf{X}_1$  and  $\mathbf{X}_2$ , respectively. The combination of  $f_{\mathbf{X}_1}$  and  $f_{\mathbf{X}_2}$  is a new potential over variables  $\mathbf{X}_{1,2} = \mathbf{X}_1 \cup \mathbf{X}_2$  defined as

$$f_{\mathbf{X}_{1,2}}(\mathbf{x}_{1,2}) = \sum_{l=1}^{r_1} \sum_{m=1}^{r_2} f_{\mathbf{X}_1}^{(l)}(\mathbf{x}_1) f_{\mathbf{X}_2}^{(m)}(\mathbf{x}_2). \quad (6)$$

**Proposition 2.** The combination of two SP factorized potentials is another SP factorized potential. That is, the class of SP factorized potentials is closed under combination.

*Proof.* Notice that each summand in Equation (6) is a product of two factorized potentials, which is itself a factorized potential. Therefore, the result of the combination is a sum of factorized potentials, and that is, by definition, an SP factorized potential. □

**Definition 7** (Marginalisation of SP factorized potentials). Let

$$f_{\mathbf{Z}}(\mathbf{z}) = \sum_{l=1}^r f_{\mathbf{Z}}^{(l)}(\mathbf{z})$$

be an SP factorized potential defined for variables  $\mathbf{Z} = \{Z_1, \dots, Z_c\}$  over hyper-cubes  $\cup_{h=1}^k \Omega_{\mathbf{Z}}^h$ . The result of marginalizing out a variable  $Z_j$  from  $f_{\mathbf{Z}}$  is a new potential defined on  $\mathbf{Z} \setminus \{Z_j\}$  over the hyper-cubes  $\cup_{h=1}^{k'} \Omega_{\mathbf{Z} \setminus \{Z_j\}}^h$ :

$$\begin{aligned} f(z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_c) \\ = \sum_{l=1}^r f_{\mathbf{Z} \setminus \{Z_j\}}^{(l)}(z_1, \dots, z_{j-1}, z_{j+1}, \dots, z_n), \end{aligned}$$

where  $f_{\mathbf{Z} \setminus \{Z_j\}}^{(l)}$ ,  $l = 1, \dots, r$ , are computed according to Definition 4.

**Proposition 3.** *The class of SP factorized potentials is closed under marginalization.*

*Proof.* As an SP factorized potential is a sum of factorized potentials, marginalizing out one variable consists of marginalizing it out in each of the factorized potentials where the variable appears. From Prop. 1 we know that the result of marginalizing out one variable from a factorized potential is an SP factorized potential.  $\square$

#### 4 Inference in BNs with MoTBFs

Consider an MoTBF model with discrete variables  $\mathbf{Y} = \{Y_1, \dots, Y_d\}$  and continuous variables  $\mathbf{Z} = \{Z_1, \dots, Z_c\}$ . Probabilistic inference in such a BN can be carried out using standard propagation algorithms that rely on sum and product operations, as the SP factorized MoTBF potentials are closed under product and marginalization.

For ease of presentation and analysis of the results, we formulate the inference process for MoTBFs using the classical variable elimination algorithm. This algorithm is designed to compute the posterior distribution over a target variable  $W \in \mathbf{X}$ . It is based on sequentially eliminating the variables in  $(\mathbf{X} \setminus \mathbf{E}) \setminus \{W\}$  (according to the minimum size heuristic with one step look ahead) from the potentials containing them. The elimination of a variable from a set of potentials is carried out by *i*) combining the potentials containing the variable and *ii*) marginalizing out that variable from the result of the combination. If not stated otherwise, we shall assume that if two variables  $X_1$  and

$X_2$  have the same parent  $Z$ , then  $\Omega_Z$  is partitioned identically in the specification of the two MoTBF potentials for  $X_1$  and  $X_2$ .

The complexity of the inference process is determined by the size of the potentials constructed during inference. The next proposition gives the size (the number of factors) resulting from combining a set of potentials, according to the combination operation described in Def. 6.

**Proposition 4.** *Let  $f^{(1)}, \dots, f^{(h)}$  be  $h$  SP factorized potentials, and let  $\mathbf{Y}_i, \mathbf{Z}_i$ ,  $i = 1, \dots, h$ , be the discrete and continuous variables of each of them. Let  $\mathbf{Z} = \cup_{i=1}^h \mathbf{Z}_i = \{Z_1, \dots, Z_j\}$ , and let  $n_l$ ,  $l = 1, \dots, j$ , be the number of intervals into which the domain of  $Z_l$  is split. If  $\Omega_{Y_i}$  is the set of possible values of the discrete variable  $Y_i$ , then it holds that*

$$\begin{aligned} \text{size}(f^{(1)} \dots f^{(h)}) \leq \\ \prod_{Y_i \in \cup_{i=1}^h \mathbf{Y}_i} |\Omega_{Y_i}| \prod_{l=1}^j n_l \prod_{l=1}^h s_l \sum_{l=1}^h t_l, \end{aligned} \quad (7)$$

where  $s_l$  and  $t_l$ ,  $l = 1, \dots, h$  are, respectively, the maximum number of summands and the maximum number of factors in each summand, in potential  $f^{(l)}$  (see Definition 5).

*Proof.* The first factor is justified by the fact that for each possible configuration of the discrete variables, there is an MoTBFs parameterization corresponding to the continuous variables. The second factor comes from the definition of combination in Definition 2, and the fact that the set of possible split points is fixed for each variable. Finally, the third and fourth factors also follow directly from Definition 6.  $\square$

Proposition 4 gives an upper bound on the size of the SP factorized potentials resulting from a combination of factors. We see a significant reduction in size with respect to the bound given in (Rumí and Salmerón, 2007, Proposition 6) for the particular case of MTEs:

$$\begin{aligned} \text{size}(f^{(1)} \dots f^{(h)}) \leq \prod_{Y_i \in \cup_{i=1}^h \mathbf{Y}_i} |\Omega_{Y_i}| \prod_{l=1}^j n_l^{k_l} \prod_{l=1}^h t_l, \end{aligned} \quad (8)$$

where  $k_l$ ,  $l = 1, \dots, h$ , is the number of continuous variables in  $f^{(l)}$  and in this case,  $t_l$ ,  $l = 1, \dots, h$  is the number of exponential terms of the potential in each hyper-cube of  $f^{(l)}$ . The differences between Equations (7) and (8) lie in the last three and two terms respectively. The difference in the second terms implies a significant reduction in size, and is based on the fact that we assume that the points in which the domain of each variable is split is selected from a fixed set of points. Under such an assumption, the number of intervals involved in the domain of a variable that appears in several potentials being combined, never increases after carrying out a combination. Note that if a potential represents a conditional MoTBF, only the domain of the conditioning variables is split and thus the corresponding term in the product would be equal to 1.

Regarding the third and forth terms, in Equation (7), they give the number of factors or univariate MoTBFs stored in each resulting hyper-cube, while in Equation (8), the third factor gives the number of exponential terms in each resulting hyper-cube. Therefore, as we are assuming that the number of summands,  $m$ , is the same for every variable, the third factor in Equation (8) could be replaced by  $m^h$ , while the number of summands reported by Equation (7) would be  $(\prod_{l=1}^h s_l)(\sum_{l=1}^h t_l)m$ , which grows more slowly than  $m^h$ , except for trivial cases. For instance, combining two univariate MoTBFs, would result in a potential with  $2m$  summands, while the same operation with traditional combination of MTEs would yield a potential with  $m^2$  exponential terms.

## 5 Experimental evaluation

In order to evaluate the proposed MoTBF-based inference procedure, we used the Variable Elimination algorithm for doing inference in a set of randomly generated networks where the number of continuous variables ranges from 4 to 24 (with increments of 2). The goal is to measure the increase in efficiency obtained by the MoTBF approach in comparison to inference using classical combination and marginalization of MTEs

(Moral et al., 2001). That is, we want to measure the impact, in terms of efficiency, of using SP factorized potentials and their corresponding operations instead of traditional potentials and their operations. Note that all the methods considered here carry out exact inference, i.e. they obtain exactly the same marginals, and therefore the only differences are in terms of efficiency. The results can be extrapolated to other formalisms compatible with the MoTBF framework, like MOPs (Shenoy and West, 2011). One advantage of choosing the MTE model as strawman framework in the experiments is the availability of the Elvira software (Elvira Consortium, 2002) that implements the MTE model. As the novelties of the approach proposed in this paper concentrate on the handling of continuous variables, we have not included discrete variables in the networks.

The networks have been constructed by randomly assigning a number of parents to each variable, following a Poisson distribution with mean 2. In order to guarantee an increase in network complexity as the number of variables grows, each new network is built by adding variables to the previous one. For instance, the network with 6 variables is obtained by randomly adding 2 variables to the already generated network with 4 variables, and so on. We assumed that in each hypercube of every conditional distribution the domain of the child variable is not split (see Eq. (3)). Also, all the univariate potentials contained in the networks correspond to 1-piece and 7-terms MoTBF approximations of the standard Gaussian density (within the range  $[-2.5, 2.5]$ ) found using the procedure described in (Langseth et al., 2012) based on exponential basis functions. The domains of the variables were split in two parts, using the mid point of the interval and keeping the split point of the variable fixed for all distributions in which they appear.<sup>4</sup> This also ensures that no new hypercubes are constructed by the combination operator, thereby reducing complexity during inference.

<sup>4</sup>For the MoTBF framework this only pertains to the conditioning variables, since the domain of a head variable is not partitioned.

| #vars | M-vars | A-vars | VE[MoTBF] | VE[MTE] |
|-------|--------|--------|-----------|---------|
| 4     | 4      | 2.7    | 0.0084    | 2.730   |
| 6     | 5      | 3.1    | 0.0094    | 300.909 |
| 8     | 5      | 3.2    | 0.0048    | 342.770 |
| 10    | 5      | 3.3    | 0.0050    | 363.877 |
| 12    | 6      | 3.7    | 0.0218    | —       |
| 14    | 9      | 4.1    | 0.0482    | —       |
| 16    | 11     | 4.7    | 0.2129    | —       |
| 18    | 12     | 5.0    | 0.5462    | —       |
| 20    | 12     | 5.0    | 0.9508    | —       |
| 22    | 14     | 5.6    | 3.4464    | —       |
| 24    | 14     | 5.7    | 9.1388    | —       |

Table 1: Average run time per variable (in seconds) of the variable elimination algorithm using the MoTBF approach (VE[MoTBF]) and the same algorithm using classical combination and marginalization (VE[MTE]). #vars indicates the number of variables in the network, M-vars is the maximum number of variables in the potentials used during the variable elimination algorithm, and A-vars is the average number of variables in the above mentioned potentials.

The results are shown in Table 1, where the average time used to run the variable elimination algorithm for each variable in the network is displayed. Empty cells in the table indicate that the algorithm ran out of memory during inference in the corresponding network (2.5GB allocated). The label VE[MoTBF] refers to the proposed variable elimination algorithm using the MoTBF approach (based on SP factorized potentials), while the label VE[MTE] indicates the same algorithm, but with traditional combination and marginalization. In all the tested cases, the proposed algorithm provides a significant improvement in efficiency, which is more evident as the number of variables increases. In fact, VE[MTE] is not able to obtain any results for networks with more than 10 variables.

With the experiment described above, we have illustrated the increase in efficiency when doing MoTBF-based inference as compared to the MTE approach. However, the use of SP-factorized potentials also carry over to the traditional MTE framework; this entails lifting the restriction of having a fixed set of possible split points for the variables, allowing split

points in the domains of child variables in conditional distributions, and not being able to exploit the properties of the basis functions. In order to analyze this modified inference algorithm, we designed a second experiment in which the split points for the variables in each potential were chosen at random. Also, instead of using MTE approximations of the Gaussian density, we used randomly generated MTEs with 10 exponential terms. The results of this experiment are displayed in Table 2, where we see a significant increase in efficiency for all the network sizes, supporting the use of SP factorized potentials even for the classical MTE framework.

| #vars | M-Vars | A-Vars | VE[MTE] <sub>L</sub> | VE[MTE] |
|-------|--------|--------|----------------------|---------|
| 4     | 3      | 2.5    | 0.0408               | 25.538  |
| 6     | 5      | 2.8    | 0.0209               | 431.836 |
| 8     | 5      | 2.6    | 0.0274               | 329.008 |
| 10    | 6      | 3.1    | 0.3432               | —       |
| 20    | 9      | 4.0    | 1.7602               | —       |

Table 2: Average run time per variable (in seconds) of the lazy variable elimination algorithm (VE[MTE]<sub>L</sub>) and the same algorithm using classical combination and marginalization (VE[MTE]) with randomly generated MTE distributions and split points. #vars, M-vars and A-vars have the same meaning as in Table 1.

In both experiments, the benefits of using the new framework are significant. The gain in efficiency, however, is not only caused by postponing the operations or by keeping the potentials in a factorized form. An important contribution to the improvement is due to the fact that basis functions are defined only over one variable. In previous MTE inference algorithms exponential functions could be defined over several variables as the products were actually carried out. This involved dealing with linear functions in the exponents (represented as lists of variables and factors in the implementation) and consume a relatively large part of the run time of the inference algorithm.

## 6 Conclusions

In this paper we have developed the necessary tools for carrying out inference with MoTBFS.

The efficiency of the MoTBF-based inference procedure comes from the use of SP factorized potentials, the properties of the basis functions, and the fact that the domains of the child variables in the conditional distributions are not split. We have shown how the basic operations necessary for inference are compatible with SP factorized potentials.

The gain in efficiency with respect to the classical MTE approach has been illustrated through an experimental analysis based on a version of the variable elimination algorithm operating with MoTBFs. We have also tested the use of SP factorized potentials within the classical MTE approach. In both cases, the results of the experiments indicate that the gain in efficiency is significant.

We have only reported on experiments over networks formed by continuous variables. The impact of incorporating discrete variables is exactly the same in all the algorithms considered in this paper. We leave for future work a more extensive experimentation incorporating discrete variables.

## Acknowledgments

Work supported by a Senior Grant in the frame of the CALL UCM-EEA-ABEL-02-2009 of the Abel Extraordinary Chair (NIFS Project), by the Spanish Ministry of Economy and Competitiveness, projects TIN2010-20900-C04-02,03 and by ERDF (FEDER) funds.

## References

- Cobb, B., Shenoy, P., and Rumí, R. (2004). Approximating probability density functions with mixtures of truncated exponentials. In *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-2004)*, Perugia, Italy. In press.
- Elvira Consortium (2002). Elvira: An environment for creating and using probabilistic graphical models. In Gámez, J. and Salmerón, A., editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models*, pages 222–230.
- Langseth, H., Nielsen, T., Rumí, R., and Salmerón, A. (2012). Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53:212–227.
- Lauritzen, S. (1992). Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108.
- Lauritzen, S. and Jensen, F. (2001). Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11:191–203.
- Madsen, A. (2010). Improvements to message computation in lazy propagation. *International Journal of Approximate Reasoning*, 51:499–514.
- Madsen, A. and Jensen, F. (1999). Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113:203–245.
- Moral, S., Rumí, R., and Salmerón, A. (2001). Mixtures of truncated exponentials in hybrid Bayesian networks. In *ECSQARU’01. Lecture Notes in Artificial Intelligence*, volume 2143, pages 135–143.
- Olesen, K. (1993). Causal probabilistic networks with both discrete and continuous variables. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:275–279.
- Rumí, R. and Salmerón, A. (2007). Approximate probability propagation with mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 45:191–210.
- Shenoy, P. (2011). A re-definition of mixtures of polynomials for inference in hybrid Bayesian networks. *ECSQARU’11. Lecture Notes in Artificial Intelligence*, 6717:98–109.
- Shenoy, P. and West, J. (2011). Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52:641–657.

# Qualitative Chain Graphs and their Use in Medicine

Martijn Lappenschaar, Arjen Hommersom, and Peter J.F. Lucas

Institute for Computing and Information Sciences,

Radboud University Nijmegen, The Netherlands

Email: {mlappens, arjeh, peterl}@cs.ru.nl

## Abstract

For modelling diseases in medicine, chain graphs are more attractive than directed graphs, i.e., Bayesian networks, as they support representing interactions between diseases that have no natural direction. In particular, representation by chain graphs is preferred over Bayesian networks as they have the ability to capture equilibrium models. Using qualitative abstractions of probabilistic interactions is also of interest in this context, as these would allow focusing on patterns in the interactions rather than looking at the numerical detail, which for medical purposes is of paramount importance. So far, qualitative abstractions of probabilistic interactions have been developed only for Bayesian networks in the form of the framework of qualitative probabilistic networks. In this paper, qualitative abstractions are developed for chain graphs with the practical purpose of using these as constraints on the hyperspace of probability distributions. The usefulness of this approach is explored for disease modelling.

## 1 Introduction

In many application fields, probabilistic graphical models are seen as convenient and intuitive formalisms to capture the probabilistic independence information of a domain. Popular graphical models include undirected graphs (UGs), also called Markov networks, and acyclic directed graphs (ADGs), also called Bayesian networks (Pearl, 1988). However, for both undirected and directed graphs one meets undesirable limitations when representing independence information for an actual problem, such as from medicine. Hybrid graphs, such as chain graphs (Lauritzen and Wermuth, 1989), that contain both directed and undirected arcs offer an elegant generalisation of both Markov networks and Bayesian networks. Chain graphs have been shown to model equilibrium systems (Lauritzen and Richardson, 2002), which occur in many areas including biology, physics, chemistry, and economics. For example, human physiology contains many regulatory mechanism that ensure homeostasis, i.e., a state of equilibrium. In particular, when modelling such

processes, even when ignoring the dimension of time, Bayesian networks are not entirely suitable and more expressive models are required.

However, Bayesian networks have the advantage that both structure and parameters can be assessed from either expert knowledge, data, or both, which renders Bayesian networks whitebox rather than blackbox models. Qualitative abstractions of Bayesian networks, qualitative probabilistic networks (QPNs), provide a useful method for exploiting qualitative constraints in assessing probabilistic information. For the more expressive chain graphs, it is much more difficult to exploit human knowledge in assessing their parameters, and, as a consequence, these models are at the moment less whitebox than Bayesian networks. The aim of the research described in this paper is to come up with ways to make chain graphs more suitable as whitebox models in particular by the use of qualitative probabilistic abstractions.

While it is well known that QPN theory has its limitation when it comes to qualitative reasoning, the main reason why QPN theory is not used in actual systems, QPNs may be quite use-

ful when looked at as offering constraints when estimating a probability distribution. This, for example, allows deriving distributions over arbitrary marginals, i.e., second-order distributions (Druzdzel and van der Gaag, 1995). If an exact probability is not required, then such distributions provide insight into the domain and could, e.g., be used to make decisions.

In this paper, we will first argue why chain graphs provide a good starting point for modelling in medicine. To support a qualitative modelling approach, we will give a formal extension of QPNs based on chain graphs. We show its usefulness by semi-qualitative reasoning in a medical example, although it can be applied to any field involving a model that is represented as a chain graph.

## 2 Motivation from the medical field

As stated in the introduction, many physiological processes within the human body can be seen as causal feedback systems, in which some kind of equilibrium setpoint is maintained. In non-healthy people the equilibrium setpoint typically differs from the healthy people, but therapeutic interventions can reset the equilibrium setpoint to a state that is closer to the healthy people. The disturbance of the equilibrium of one physiological process, might also alter the equilibrium setpoints of other regulation systems, which might in turn induce new pathophysiology and decrease the patient’s prognosis even further. In the interest of the physician, it is important to know the qualitative dynamics of such interactions, i.e., is it more likely that a therapy for a specific disease give rise to symptoms of another (patho)physiological process. Some of these aspects are illustrated in the following example, which will be used as a running example throughout the paper.

**Example 1.** Figure 1 shows an abstraction of the interaction between two diseases, i.e., diabetes mellitus and lipid disorder, along with its typical blood measurements, a risk factor, i.e., obesity, and a possible therapy for diabetes. It is assumed that there is feedback between the pathophysiology of both diseases, which is al-

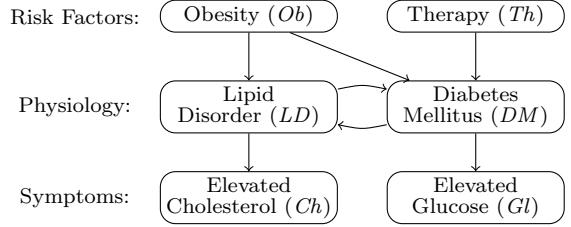


Figure 1: Schematic representation of an interaction between diabetes mellitus and a lipid disorder, showing that between the diseases feedback exists within their pathophysiology.

most always in some kind of equilibrium. The association between these pathophysiologies can be measured by the fact that the symptoms are associated, i.e., elevated glucose levels are associated with elevated cholesterol levels.

## 3 Preliminaries

### 3.1 Chain Graphs

A chain graph (CG) is a probabilistic graphical model that consists of labelled vertices, representing random variables, connected by directed and undirected edges. The definitions here are in accordance with existing literature on chain graphs (Studený and Bouckaert, 1998).

Let  $G = (V, E)$  be a *hybrid graph*, where  $V$  denotes the set of *vertices* and  $E$  the set of *edges*, where an edge is either an *arc* (directed edge), or a *line* (undirected edge). Let indexed letters, e.g.,  $V_1$  and  $V_2$ , indicate vertices of a chain graph. We denote an arc connecting two vertices by ‘ $\rightarrow$ ’ and a line by ‘ $-$ ’. Consider two vertices  $V_1$  and  $V_2$ . If  $V_1 \rightarrow V_2$  then  $V_1$  is a *parent* of  $V_2$ . If  $V_1 - V_2$  then  $V_1$  is a *neighbour* of  $V_2$ . The set of parents and neighbours of a vertex  $V_i$  are denoted by  $\text{pa}(V_i)$  and  $\text{ne}(V_i)$ , respectively. The set  $\text{pa}(V_i) \cup \text{ne}(V_i)$  is the *boundary* of  $V_i$ , denoted by  $\text{bd}(V_i)$ . We will denote  $\text{cl}(V_i)$  as the *closure* of  $V_i$  defined by  $\text{bd}(V_i) \cup \{V_i\}$ .

A *path* of length  $n$  in a hybrid graph  $G$  is a sequence of distinct vertices  $V_1, \dots, V_{n+1}$ , such that either  $V_i - V_{i+1} \in E$ ,  $V_i \rightarrow V_{i+1} \in E$ , or  $V_i \leftarrow V_{i+1} \in E$ . A *directed path* is a path which includes at least one arc, and where all arcs have the same direction. A *chain graph* is a hybrid graph with the restriction that no

directed cycles exist. A *descending path* is a path, where there are no  $V_i \leftarrow V_{i+1} \in E$ . A vertex  $V_i$  is an ancestor of  $V_j$  if there exists an descending path from  $V_i$  to  $V_j$ . The set  $\text{an}(V_i)$  denotes the set of *ancestors* of vertices from  $V_i$ .

If there is a line between every pair of vertices in a set of vertices, then this set is named *complete*. A *clique* is a maximally complete subset. Removing all the arcs from the graph leaves us with vertices connected by lines, called *chain components*; the set of all chain components is denoted here by  $\mathcal{C}$ . The *family* of a vertex  $V_i$ , denoted by  $\text{fa}(V_i)$ , is the set  $C \cup \text{pa}(C)$  where  $C \in \mathcal{C}$  and  $V_i \in C$ .

Associated to a chain graph  $G = (V, E)$  is a joint probability distribution over the set of vertices  $P(V)$  that is faithful to the chain graph  $G$ , i.e., it includes all the independencies implied by the graph. In this paper, we assume  $P(V)$  to be a strictly positive discrete factorisable distribution, which are almost always faithful (Peña, 2009), defined by an *outer factorisation*:

$$P(V) = \prod_{C \in \mathcal{C}} P(C \mid \text{pa}(C)) \quad (1)$$

where each  $P(C \mid \text{pa}(C))$  is defined by a clique-wise factorisation:

$$P(C \mid \text{pa}(C)) = Z^{-1}(\text{pa}(C)) \prod_{M \in M_C} \varphi_M(M) \quad (2)$$

given that  $M_C$  are the complete subsets in the closure graph of  $C$ , i.e., the subgraph  $G_{C \cup \text{pa}(C)}$  where each arc is replaced by a line and each distinct vertex of  $\text{pa}(C)$  is also connected by a line. The functions  $\varphi$  are non-negative real functions, called *potentials*; they generalise joint probability distributions in the sense that they do not need to be normalised. The constant  $Z(\text{pa}(C)) = \sum_C \prod_{M \in M_C} \varphi_M(M)$  normalises the product to a probability distribution.

Undirected edges in chain graphs can be interpreted as an equilibrium (steady-state) in a feedback model (Lauritzen and Richardson, 2002). For example, consider again the graph in Figure 1, where there is a feedback relationship between lipid disorder and diabetes mellitus. In practice, this feedback system is in a steady

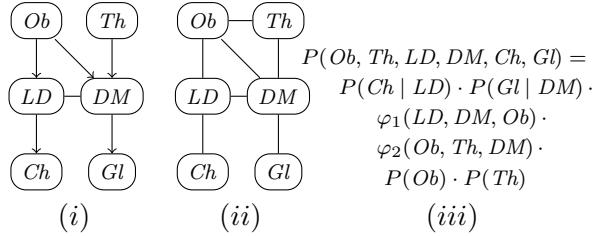


Figure 2: Chain graph representation (i), closure graph of chain components (ii), and factorisation (iii) of the example in Figure 1.

state, although the setpoint of the feedback system may be changed, for example the amount of insulin resistance. Therefore, only the relationships between variables within a steady-state are relevant, rather than the underlying dynamic process that leads to the equilibrium. Moreover, the underlying dynamics is very difficult to measure *in vivo*, hence, the parameters of such models are difficult to elicit. Therefore, we argue that chain graph offer an attractive abstraction of the underlying dynamic mechanism for feedback systems in disease models. The corresponding chain graph and factorisation of Figure 1 in its steady state is shown in Figure 2.

### 3.2 QPNs

*Qualitative probabilistic networks* (QPNs) were introduced by Wellman (1990), as a qualitative abstraction of Bayesian networks. Conditional probability distributions are replaced by qualitative knowledge in the form of signs, which describes the relationships among variables by the concepts of probabilistic influences and synergies. Here we briefly recall the theory in accordance with the definitions of Renooij (2001).

For clarity of exposition, we will assume that each node  $A \in V$  in the discrete chain graph model is a binary variable, which can take the values  $a$  ( $A = \text{true}$ ) and  $\bar{a}$  ( $A = \text{false}$ ). Further, for notational convenience, we will sometimes write the singleton set  $\{A\}$  as  $A$ , and, if  $X, Y \subseteq V$ , then we will write  $XY$  instead of  $X \cup Y$ . Finally, we denote  $X - Y$  for  $X \setminus Y$ . For example  $X - AB$  is an abbreviation of  $X \setminus \{A, B\}$ .

Then, a *qualitative influence* expresses how the value of one variable influences the probabil-

ity of observing values of another variable. Let  $Z$  denotes the set of variables  $\text{pa}(B) - A$ . We say that  $A$  has a *positive qualitative influence on  $B$* , if  $P(b | a, z) - P(b | \bar{a}, z) \geq 0$ , regardless of the configuration  $z$ , with a strict inequality for at least one configuration  $z$ .

An *additive synergy* expresses how the interaction between two variables influences the probability of observing the values of a third variable. Now, let  $Z$  denotes the set consisting the variables  $\text{pa}(B) - A_1 A_2$ . We say there is a *positive additive synergy of  $A_1$  and  $A_2$  on  $B$* , if  $P(b | a_1, a_2, z) + P(b | \bar{a}_1, \bar{a}_2, z) - P(b | \bar{a}_1, a_2, z) - P(b | a_1, \bar{a}_2, z) \geq 0$ , regardless of the configuration  $z$ , with a strict inequality for at least one configuration  $z$ .

A *product synergy* expresses how upon observation of a common child of two vertices, observing the value of one parent vertex influences the probability of observing a value of the other parent. We say there is a *positive product synergy of  $A_1$  and  $A_2$*  with regard to the value  $b$  on variable  $B$ , if  $P(b | a_1, a_2, z) \cdot P(b | \bar{a}_1, \bar{a}_2, z) - P(b | \bar{a}_1, a_2, z) \cdot P(b | a_1, \bar{a}_2, z) \geq 0$ , regardless of the configuration  $z$ , with a strict inequality for at least one configuration  $z$ .

*Negative* and *zero* influences and synergies are defined analogously, by replacing  $\geq$  with  $\leq$  and  $=$  respectively. If none of these cases hold, we say that the influence or synergy is *ambiguous*.

## 4 Qualitative Chain Graphs

In this section, we will analyse influences and synergies in the context of chain graph models.

### 4.1 Influences in chain graphs

The properties of signs in qualitative probabilistic networks rely on the fact that signs hold in any context, i.e., intuitively, a variable  $A$  positively influences another variable  $B$  if in any possible context the probability of  $B$  is higher for  $a$  compared to  $\bar{a}$ . While such a context is relatively clear in case of directed arcs, it is more subtle for probabilistic chain graphs, in which influences can also exist through lines. To obtain a proper definition in such a network, we will define influences in terms of *interventions* (Lauritzen and Richardson, 2002) on particular variables in the chain graph.

**Definition 1.** The influence of  $A$  on  $B$  in a context  $c \in V - AB$ , where  $A$  and  $B$  are two vertices, is the probability  $P(b \parallel a, c) - P(b \parallel \bar{a}, c)$  where  $P(B \parallel A, C)$  is the probability of  $B$  after an intervention on  $A$  and  $C$ .

We say that  $A$  has a positive influence on  $B$  if the influence of  $A$  on  $B$  is  $\geq 0$  in any context. Negative, zero and ambiguous influences are defined similarly. Similarly to QPNs, these influences coincide with a difference in conditional probabilities if we assume that the chain graph can be given a causal interpretation and the chain components model equilibria, which will be assumed in the remainder of this paper.

**Lemma 1.** *If a chain graph  $G = (V, E)$  is generated by a causal feedback model where lines represent equilibria (Lauritzen and Richardson, 2002), and  $P(B \parallel V - B)$  denotes the probability distribution of  $B$  after an intervention on all other variables, then:*

$$P(B \parallel V - B) = P(B \mid \text{fa}(B) - B)$$

*Proof.* This is a direct corollary of Equation (18) in Lauritzen and Richardson (2002).  $\square$

Given the fact that a node is independent of its non-descendants given its boundary, i.e., for models that factorise as given in Section 3.1, a local Markov property holds, which yields the following lemma.

**Lemma 2.** *Given a chain graph  $G = (V, E)$  and vertices  $A, B \in V$  such that  $A \in \text{bd}(B)$ , then  $P(B \mid \text{an}(B) - B) = P(B \mid \text{fa}(B) - B) = P(B \mid \text{bd}(B))$ .*

*Proof.* Follows from the local Markov property of chain graphs (Frydenberg, 1990):  $V_i \perp\!\!\!\perp \text{an}(V_i) - \text{cl}(V_i) \mid \text{bd}(V_i)$ .  $\square$

Then, using Lemmas 1 and 2, we obtain an expression of influences in chain graphs in terms of conditional probabilities.

**Proposition 1.** *Given two nodes  $A$  and  $B$  and a context  $c$ , then the influence of  $A$  on  $B$  in context  $c$  equals:*

$$P(b \mid a, z) - P(b \mid \bar{a}, z)$$

where  $c = z \cup x$ ,  $Z = \text{bd}(B) - A$ , and  $X = V - ZAB$ .

Note that it follows that the influence of node  $A$  on node  $B$  is a zero influence if  $A \notin \text{bd}(B)$  and  $A \in \text{an}(B)$ . Also note that the qualitative influences generalise the QPN definitions, since  $\text{bd}(B) = \text{pa}(B)$  for any  $B \in V$  if every chain components consist of a single vertex.

## 4.2 Symmetry of influences

In QPNs, the qualitative signs are symmetric, i.e., if there is some influence from a node  $A$  to a node  $B$ , then there is an influence from  $B$  to  $A$  with the same sign. Therefore, only a single sign is needed for every arc in a QPN. In the remainder, we will prove that this symmetry is preserved for qualitative chain graphs, i.e., also for neighbouring nodes the signs are symmetric. First we prove a lemma that rephrases qualitative influences in terms of relationships between potential functions. For notational convenience, we do not write universally quantified variables in potentials, e.g.,  $\varphi_M(a)$  is shorthand for  $\varphi_M(M - A, a)$ , and we will write  $\varphi_M(a, X)$  for  $\varphi_M(X)$  if  $A \notin M$  for any  $X \subseteq V$ . Further, we will focus in the next lemma and theorem on positive influences, however, the same reasoning holds for negative and zero influences.

**Lemma 3.** *Given a chain graph  $G$  containing vertices  $A$  and  $B$ , with  $A \in \text{bd}(B)$  and  $B$  an element of a component  $C$ , it holds that:*

$$P(b | a, \text{fa}(B) - AB) \geq P(b | \bar{a}, \text{fa}(B) - AB)$$

if and only if

$$\prod_{M \in M_{AB}} \varphi_M(a, b) \varphi_M(\bar{a}, \bar{b}) \geq \prod_{M \in M_{AB}} \varphi_M(a, \bar{b}) \varphi_M(\bar{a}, b)$$

where  $M_{AB} = \{M \in M_C \mid \{A, B\} \subseteq M\}$ .

*Proof.* By basic probability theory, we have:

$$\begin{aligned} P(B | A, \text{fa}(B) - AB) &= \frac{P(C)}{P(C - B)} \\ &= \frac{P(C | \text{pa}(C)) P(\text{pa}(C))}{\sum_B P(C | \text{pa}(C)) P(\text{pa}(C))} \end{aligned}$$

Using Equation (2), that factorises conditional probabilities of a component into potentials, the left-hand side therefore equals to:

$$\begin{aligned} \frac{Z^{-1}(\text{pa}(C)) \left( \prod_{M_C} \varphi_M(a, b) \right) P(\text{pa}(C))}{\sum_B Z^{-1}(\text{pa}(C)) \left( \prod_{M_C} \varphi_M(a, B) \right) P(\text{pa}(C))} &\geq \\ \frac{Z^{-1}(\text{pa}(C)) \prod_{M_C} (\varphi_M(\bar{a}, b)) P(\text{pa}(C))}{\sum_B Z^{-1}(\text{pa}(C)) \left( \prod_{M_C} \varphi_M(\bar{a}, B) \right) P(\text{pa}(C))} \end{aligned}$$

Given that  $B \in C$ , we have  $B \notin \text{pa}(C)$ , so the term  $Z^{-1}(\text{pa}(C))P(\text{pa}(C))$  only depends on  $A$ . By replacing this term by  $f(A)$  and multiplying each side by the denominators, we obtain:

$$\begin{aligned} \prod_{M_C} \varphi_M(a, b) f(a) \sum_B \prod_{M_C} \varphi_M(\bar{a}, B) f(\bar{a}) &\geq \\ \prod_{M_C} \varphi_M(\bar{a}, b) f(\bar{a}) \sum_B \prod_{M_C} \varphi_M(a, B) f(a) \end{aligned}$$

Writing out the possible values for the summation over  $B$ , i.e.,  $b$  and  $\bar{b}$ , we obtain:

$$\begin{aligned} \prod_{M_C} \varphi_M(a, b) \varphi_M(\bar{a}, b) f(a) f(\bar{a}) + \\ \prod_{M_C} \varphi_M(a, b) \varphi_M(\bar{a}, \bar{b}) f(a) f(\bar{a}) &\geq \\ \prod_{M_C} \varphi_M(\bar{a}, b) \varphi_M(a, b) f(a) f(\bar{a}) + \\ \prod_{M_C} \varphi_M(\bar{a}, b) \varphi_M(a, \bar{b}) f(a) f(\bar{a}) \end{aligned}$$

Removing the factors that are the same on both sides of the equation we get:

$$\prod_{M_C} \varphi_M(a, b) \varphi_M(\bar{a}, \bar{b}) \geq \prod_{M_C} \varphi_M(a, \bar{b}) \varphi_M(\bar{a}, b)$$

For all potentials not depending on both  $A$  and  $B$ , e.g.,  $\varphi_M(A, B) = \varphi_M(B)$ , its corresponding factors are also the same on both sides of the equation, leaving us with:

$$\prod_{M \in M_{AB}} \varphi_M(a, b) \varphi_M(\bar{a}, \bar{b}) \geq \prod_{M \in M_{AB}} \varphi_M(a, \bar{b}) \varphi_M(\bar{a}, b)$$

□

|            | $ld, dm$ | $\bar{ld}, dm$ | $ld, \bar{dm}$ | $\bar{ld}, \bar{dm}$ |
|------------|----------|----------------|----------------|----------------------|
| $ob$       | 16       | 4              | 2              | 4                    |
| $\bar{ob}$ | 2        | 2              | 1              | 5                    |

 Table 1: Potentials over  $Ob$ ,  $LD$ , and  $DM$ .

**Example 2.** Continuing Example 1, to evaluate a (say, positive) influence of  $Ob$  on  $LD$  only involves  $\varphi_1$  (cf. Figure 1c). Therefore, a positive influence of  $Ob$  on  $LD$  is equivalent to:

$$\begin{aligned} & \varphi_1(ob, ld, DM)\varphi_1(\bar{ob}, \bar{ld}, DM) \\ & \geq \varphi_1(\bar{ob}, ld, DM)\varphi_1(ob, \bar{ld}, DM) \end{aligned}$$

for all values of  $DM$ . Consider, as an example, the potential defined in Table 1. It holds for  $dm \rightarrow 16 \geq 4$ , and for  $\bar{dm} \rightarrow 20 \geq 8$ , implying a positive influence of  $Ob$  on  $LD$ . Indeed, by computing the individual probabilities using Table 1, we obtain:

$$\begin{aligned} P(ld | ob, dm) &= 8/9 \geq P(ld | \bar{ob}, dm) = 2/3 \\ P(ld | ob, \bar{dm}) &= 2/3 \geq P(ld | \bar{ob}, \bar{dm}) = 4/9 \end{aligned}$$

As a result of this lemma, determining the nature of a qualitative influence between two vertices implies that one only has to consider those potentials for cliques containing the two variables that describe the influence. Hence, we have the following result that we were aiming for, proving the symmetry between qualitative influences between arbitrary edges.

**Theorem 1.** *It holds that qualitative signs of chain graphs are symmetric, i.e., suppose  $(A, B) \in E$ , then  $P(b | a, X) - P(b | \bar{a}, X) \geq 0$  if and only if  $P(a | b, Y) - P(a | \bar{b}, Y) \geq 0$ , where  $X = \text{bd}(B) - A$  and  $Y = \text{bd}(A) - B$ .*

*Proof.* ( $\Rightarrow$ ) Assume  $P(b | a, X) - P(b | \bar{a}, X) \geq 0$ . Since  $Y \subseteq \text{an}(B) - \text{cl}(B)$ , by the local Markov property, we have  $B \perp\!\!\!\perp Y | AX$ , so it follows that  $P(b | a, X, Y) - P(b | \bar{a}, X, Y) \geq 0$ . By Proposition 1 and Lemma 3, it follows that

$$\prod_{M \in M_{AB}} \varphi_M(a, b) \varphi_M(\bar{a}, \bar{b}) \geq \prod_{M \in M_{AB}} \varphi_M(a, \bar{b}) \varphi_M(\bar{a}, b).$$

By the same reasoning, we get  $P(a | b, x, y) - P(a | \bar{b}, x, y) \geq 0$  and given that  $A \perp\!\!\!\perp X | BY$ ,

we conclude  $P(a | b, y) - P(a | \bar{b}, y) \geq 0$ . The converse holds by the same steps.  $\square$

### 4.3 Qualitative chain graphs defined

Given the properties of influences in chain graphs, we are now in the position to define the usual notions of qualitative probabilistic networks for chain graphs. We will focus on the positive influences and synergies; the negative, zero and ambiguous influences and synergies are defined similarly.

**Definition 2.** We say that a vertex  $A$  positively influences a vertex  $B$ , written as  $S^+(A, B)$ , iff  $A \in \text{bd}(B)$  and

$$P(b | a, \text{bd}(B) - A) \geq P(b | \bar{a}, \text{bd}(B) - A)$$

A positive *additive synergy* expresses that the joint influence of  $A_1$  and  $A_2$  is greater (or less in case of a negative synergy) than their separate influence on a child  $B$ . More formally, we say that influences of  $A_1$  on  $B$  are higher in contexts  $c$  where  $a_2 \in c$  compared to contexts  $c'$  where  $\bar{a}_2 \in c$ . This can be rephrased in terms of conditional probabilities following Proposition 1.

**Definition 3.** We say that vertices  $A_1$  and  $A_2$  express a positive additive synergy on a vertex  $B$ , written as  $Y^+(\{A_1, A_2\}, B)$ , iff  $A_1, A_2 \in \text{bd}(B)$ ,  $Z = \text{bd}(B) - A_1 A_2$ , and

$$\begin{aligned} P(b | a_1, a_2, Z) - P(b | \bar{a}_1, a_2, Z) &\geq \\ P(b | a_1, \bar{a}_2, Z) - P(b | \bar{a}_1, \bar{a}_2, Z) \end{aligned}$$

A *product synergy* expresses how the value of one cause influences the probability of the value of another cause when observing the common child the child. By similar reasoning for influences and additive synergies, we define product synergies as follows.

**Definition 4.** We say that vertices  $A_1$  and  $A_2$  express a negative product synergy with regard to the value  $b$  on the vertex  $B$ , written as  $X^+(\{A_1, A_2\}, b)$ , iff  $A_1, A_2 \in \text{bd}(B)$ ,  $Z = \text{bd}(B) - A_1 A_2$ , and

$$\begin{aligned} P(b | a_1, a_2, Z) \cdot P(b | \bar{a}_1, \bar{a}_2, Z) &\geq \\ P(b | a_1, \bar{a}_2, Z) \cdot P(b | \bar{a}_1, a_2, Z) \end{aligned}$$

## 5 Experimental Results

Probabilistic inference with a QPN can be done using sign-propagation, based on message-passing between neighbouring nodes (Druzdzel and Henrion, 1993), which has its limitations in case of trade-offs. An alternative approach is to look upon the qualitative signs as constraints on the joint probability distribution, as proposed in Druzdzel and van der Gaag (1995), where a canonical representation consisting of (in)equalities expressing constraints on the hyperspace of possible joint probability distributions is used. In this approach, some of the conditional probabilities or cliques may be elicited from experts or learned from data, where for others, only qualitative information is available.

In this paper, we take a similar approach, where we sample the unknown potentials from the factorisation of a given chain graph (cf. Equations (1) and (2)). Instead of sampling the full joint probability distributions and then establishing if the distribution is consistent with qualitative influences, the potentials can be sampled more efficiently by Lemma 3, as this shows that influences impose *local* constraints on the potentials. Likewise, synergies can be stated in terms of constraints on the local potentials using the following proposition.

**Proposition 2.** *Given a chain graph  $G$  containing vertices  $A_1$ ,  $A_2$ , and  $B$ , with  $A_1, A_2 \in \text{bd}(B)$  and  $B$  an element of a component  $C$ , it holds that a positive additive synergy  $Y^+(\{A_1, A_2\}, B)$  exists if and only if*

$$\begin{aligned} \phi_C(a_1, a_2, b) + \phi_C(\bar{a}_1, \bar{a}_2, b) &\geq \\ \phi_C(a_1, \bar{a}_2, b) + \phi_C(\bar{a}_1, a_2, b) & \end{aligned}$$

and, likewise, a positive product synergy  $X^+(\{A_1, A_2\}, B)$  exists if and only if

$$\begin{aligned} \phi_C(a_1, a_2, b) \cdot \phi_C(\bar{a}_1, \bar{a}_2, b) &\geq \\ \phi_C(a_1, \bar{a}_2, b) \cdot \phi_C(\bar{a}_1, a_2, b) & \end{aligned}$$

with  $\phi_C(a_1, a_2, b) = \frac{\prod_{M \in M_{A_1 A_2 B}} \varphi(a_1, a_2, b)}{\sum_B \prod_{M \in M_{A_1 A_2 B}} \varphi(a_1, a_2, B)}.$

---

**Procedure 1** *sample-distribution*(potentials  $\phi_{known}$ ,  $\phi_{unknown}$ , qualitative constraints  $C$ )

---

```

for $\phi_M \in \phi_{unknown}$ do
 $\phi_M \leftarrow$ sample a potential for variables M
 while !satisfies1(ϕ_M, C) do
 $\phi_M \leftarrow$ resample potential for M
 end while
 $\phi_{known} \leftarrow \phi_{known} \cup \{\phi_M\}$
end for
return distribution using Equations 1 and 2

```

---

The proof of Proposition 2 follows the same line as the proof of Lemma 3, but is omitted here due to lack of space.

Given these properties, distributions can be sampled that satisfy the qualitative constraints (cf. Procedure 1). Then, using these samples, second-order distributions of arbitrary marginal distributions can be derived in a straightforward manner. While typically the marginals range over the whole  $[0, 1]$  interval, the qualitative constraints alter the shape (e.g., the mean and variance) of the distribution, which can then be used to draw conclusions from the model.

**Example 3.** Continuing from Example 1, consider the quantitative and qualitative information available in Figure 3a. The 2<sup>nd</sup> order distribution of *Ch*, i.e., high cholesterol, is then shown in Figure 3b. An intervention on *Th*, yields the 2<sup>nd</sup> order distribution in Figure 3c, showing that probability of *Ch* shifts to lower values. The probability  $P(Ch | Th) < P(Ch) \approx 0.82$  within the generated samples ( $n=100.000$ ), suggesting with high confidence that diabetic therapy is also beneficial to reduce cholesterol levels. Note that it has been derived without any quantitative information about the chain component containing *LD* and *DM*. An additional positive synergy between *Ob* and *Th* on *DM* pushes the distribution even more to lower probabilities with an intervention on *Ob*, see Figure 3d. The probability  $P(Ch | \overline{Ob}, Th) < P(Ch) \approx 0.91$ , suggesting that an additional reduction of weight in combination with diabetic therapy is even more beneficial to reduce cholesterol levels.

---

<sup>1</sup>By local constraints of Lemma 3 and Proposition 2.

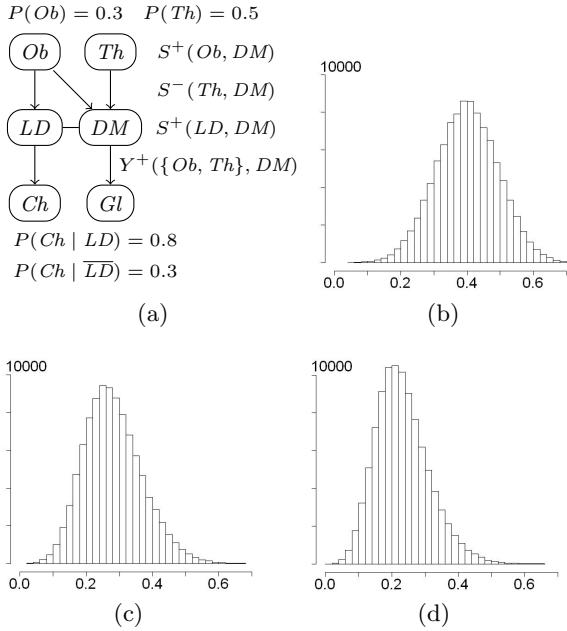


Figure 3: Qualitative and quantitative information (a) of Figure 2(i), and 2<sup>nd</sup> order probability distributions of *Ch* (b,c,d), in the presence of specific interventions (see Example 3).

## 6 Conclusions

In this paper we extended the QPN framework, i.e., qualitative influences and synergies, towards chain graphs. We analysed influences in chain graphs and showed some of its basis properties. This allows the modelling of qualitative representation of feedback systems, e.g., (patho)physiological processes, since chain graphs have the ability to model equilibria of such systems. We have illustrated that within the same chain graph it is feasible to combine quantitative and qualitative information.

This is of importance for medicine, i.e., without knowing the exact joint probability distribution that exists between diseases, we are still able to draw qualitative conclusions on the dynamics that exist within a disease model. Starting from quantitative information, e.g., conditional probabilities relating symptoms and diseases, adding qualitative information, can be efficiently done by putting specific constraints on the potentials of local cliques in the chain graph. Such information often comes in terms

of so-called odds ratios, which easily translates to qualitative influence and synergies.

In future work we aim to apply this formalism in a study on diabetes and cardiovascular comorbidities involving multiple feedback systems. Furthermore, the sampling of potentials may be improved by exploiting Monte Carlo methods which take into account bounds on the hyperspace (e.g. based on (Smith, 1984)).

## References

- MJ Druzdzel and M Henrion. 1993. Efficient reasoning in qualitative probabilistic networks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, California. AAAI Press.
- MJ Druzdzel and LC van der Gaag. 1995. Elicitation of probabilities for belief networks: Combining qualitative and quantitative information. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 141–148.
- M. Frydenberg. 1990. The chain graph Markov property. *Scand J Statist*, 17:333–353.
- SL Lauritzen and TS Richardson. 2002. Chain graph models and their causal interpretations. *J.R. Statist. Soc. B*, 64(3):321–361.
- SL Lauritzen and N Wermuth. 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1):31–57.
- J Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.
- JM Peña. 2009. Faithfulness in chain graphs: The discrete case. *J. of Approximate Reasoning*, 50(8):1306–1313.
- S Renooij. 2001. *Qualitative Approaches to Quantifying Probabilistic Network*. Ph.D. thesis, Utrecht University.
- RL Smith. 1984. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308.
- M Studený and RR Bouckaert. 1998. On chain graph models for description of conditional independence structures. *The Annals of Statistics*, 26(4):1434–1495.
- MP Wellman. 1990. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 40:257–303.

# A Depth-First Search Algorithm for Optimal Triangulation of Bayesian Network

Chao Li

University of Electro-Communications, Japan

ricyou@ai.is.uec.ac.jp

Maomi Ueno

University of Electro-Communications, Japan

ueno@ai.is.uec.ac.jp

## Abstract

Finding the triangulation of a Bayesian network with minimum total table size reduces the computational cost for probabilistic reasoning in the Bayesian network. This task can be done by conducting a search in the space of all possible elimination orders of the Bayesian network. However, such a search is known to be NP-hard. To relax this problem, Ottosen and Vomlel (2010b) proposed a depth-first branch and bound algorithm, which reduces the computational complexity from  $\Theta(\beta \cdot |V|!)$  to  $O(\beta \cdot |V|!)$ , where  $\beta$  describes the overhead computation per node in the search space, and where  $|V|!$  is the search space size. Nevertheless, this algorithm entails a heavy computational cost. To mitigate this problem, this paper presents a proposal of an extended algorithm with the following features: (1) Reduction of the search space to  $O((|V|-1)!)$  using Dirac's theorem, and (2) reduction of the computational cost  $\beta$  per node. Some simulation experiments show that the proposed method is consistently faster than Ottosen and Vomlel's method.

## 1 Introduction

The most influential method for exact inference in Bayesian networks is the join tree propagation algorithm (Jensen et al., 1990), which works in two steps: compilation and propagation. The compilation part of the method

- triangulates the graph (i.e., add extra fill-in edges such that every cycle of length greater than three has a chord),
- forms a factor  $\Phi_C$  for each clique  $C$  of the triangulated graph, and
- constructs a junction tree over these cliques.

The propagation part of the method passes messages in the whole junction tree. In the Hugin architecture, the message passing can be organized so that one passing in each direction

of the links of the junction tree. The computational cost for each message computation is proportional to the factor table size of clique. To reduce the computational cost, we need to find a triangulation with minimum total table size, which is the summation of the factor table sizes of all cliques.

Heuristic algorithms include popular triangulation heuristics like min-degree, and min-fill is wildly used for triangulation (Darwiche, 2009), but these heuristics do not guarantee the exact solution.

Although finding the optimal triangulation can be conducted by searching all possible elimination orders, the problem is known to be NP-hard (Wen, 1990). Fortunately, this is not a critical defect because triangulation can often be done off-line on specialized servers. Moreover, the triangulation results can be saved for inference algorithms. To tackle the optimal triangulation, Gogate and Dechter (2004) used

depth-first search and Dow and Korf (2007) used best-first search. However, they used treewidth criterion, which can not guarantee the minimum TTS.

To obtain the exact solution for TTS criterion, Ottosen and Vomlel (2010b) proposed a branch and bound algorithm. This algorithm uses TTS of the current partially triangulated graph as a lower bound. To compute this lower bound for each node on the search space, the cliques of the current graph must be calculated. For this purpose, Ottosen and Vomlel (2010a) proposed a dynamic clique maintenance algorithm. Consequently, they reduced the computational complexity from  $\Theta(\beta \cdot |V|!)$  to  $O(\beta \cdot |V|!)$ , where  $\beta$  describes the per-node overhead computation and where  $|V|!$  is the size of the search space. However, this method still entails a heavy computational cost.

To relax this problem, based on Ottosen and Vomlel (2010b), this paper presents a proposal of a depth-first search with the following features:

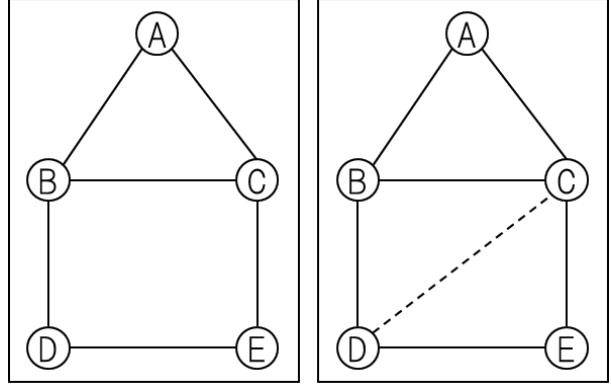
1. Reduction of search space using Dirac's theorem to  $O((|V|-1)!)$ , and
2. reduction of the computational cost  $\beta$  per node.

Some simulation experiments show that the proposed method is consistently faster than Ottosen and Vomlel's method.

## 2 Preliminaries

We shall use the following definitions and notations according to Ottosen and Vomlel (2010b).  $G = (V, E)$  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  $\{u, v \mid (u, v) \in F\}$ . For a set of vertices  $W \subseteq V$ ,  $G[W]$  is the subgraph of  $G$  that is induced by  $W$ .

Two vertices  $u$  and  $v$  are connected in  $G$  if an edge exists 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. If  $W$  is complete and no complete set  $U$  exists such that  $W$



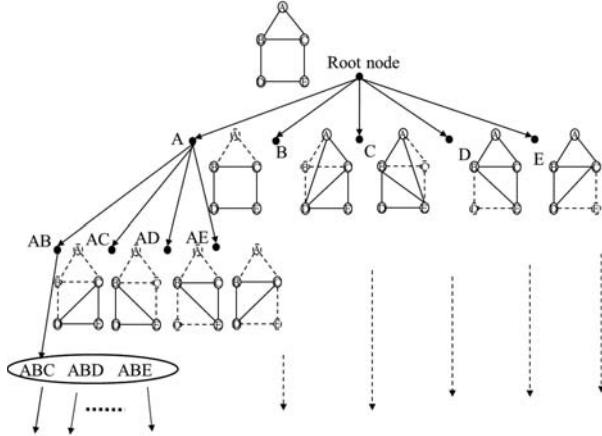
**Figure 1:** Left: Initial graph  $G = (V, E)$ . Right: Updated graph  $G'$  by adding one edge  $\{C, D\}$ . We have  $\mathcal{C}(G) = \{\{A, B, C\}, \{B, D\}, \{D, E\}, \{C, E\}\}$  and  $\mathcal{C}(G') = \{\{A, B, C\}, \{B, C, D\}, \{C, D, E\}\}$ . In this example, we have  $\mathcal{RC}(G, G') = \{\{B, D\}, \{D, E\}, \{C, E\}\}$  and  $\mathcal{NC}(G, G') = \{\{B, C, D\}, \{C, D, E\}\}$ .

$\subset U$ , then  $W$  is a *clique*. (Remark: Any complete set is sometimes called a clique. Therefore, what we defined as a clique is called a maximal clique.) The set of all cliques of a graph is denoted as  $\mathcal{C}(G)$ . The set of all cliques that intersect a set of vertices  $W$  is denoted as  $\mathcal{C}(W, G)$ .

The neighbors  $\text{nb}(v, G)$  of a vertex  $v \in V$  is the set  $W \subset V$  such that each  $u \in W$  is connected to  $v$ . The neighbors  $\text{nb}(W, G)$  of a set of vertices  $W$  are those vertices from  $V \setminus W$ , which are connected to at least one vertex  $v \in W$ . The *family*  $\text{fa}(W, G)$  of a set of vertices  $W$  is the set  $\text{nb}(W, G) \cup W$ .

If  $G' = (V, E \cup F)$  is the graph resulting from adding a set of new edges  $F$  to  $G$ , then  $\mathcal{RC}(G, G') = \mathcal{C}(G) \setminus \mathcal{C}(G')$  is the set of removed cliques, and  $\mathcal{NC}(G, G') = \mathcal{C}(G') \setminus \mathcal{C}(G)$  is the set of new cliques. Figure 1 presents these concepts.

An undirected graph is *triangulated* (or chordal) if every cycle of length four or more has a chord that is an edge of the graph joining two non-adjacent vertices of the cycle. 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  $\mathcal{T}(G)$ . For example, in Figure 1, the graph on the left is not triangulated because there exists a cycle  $\{B, C, D, E\}$  of length four in which a chord does not exist. The graph on the



**Figure 2:** Search tree for exploring all elimination orders of the graph.

right is triangulated because the edges  $\{\{C,D\}\}$  is a triangulation T for the graph on the left.

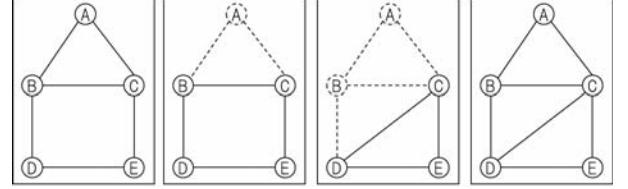
The elimination of a vertex  $v \in V$  of  $G = (V, E)$  is the process of removing v from G and making  $\text{nb}(v, G)$  a complete set. This process induces a partially triangulated graph  $H = (V, E \cup F)$ , where F is the set of fill-in edges. If  $F = \emptyset$ , then v is a simplicial vertex. An *elimination order*  $\pi = \{v_1, v_2, \dots, v_n\}$  is a permutation of vertices specifying an ordering for eliminating all vertices V of G. A prefix  $\tau$  of an elimination order  $\pi$  is a sequence of vertices that occurs at the beginning of  $\pi$ . If all vertices are eliminated in G according to an elimination order  $\pi$ , then the union of all the fill-in edges induces a triangulation T of G. Consequently, each triangulation T of G corresponds to at least one elimination order. We can explore the space  $\mathcal{T}(G)$  by investigating all possible elimination orders.

The table size of a clique C is given as  $ts(C) = \prod_{v \in C} |\text{sp}(v)|$ , where  $\text{sp}(v)$  denotes the state space of the variable corresponding to v in the Bayesian network. Finally, the TTS of a graph H is given as  $tts(H) = \sum_{C \in \mathcal{C}(H)} ts(C)$ .

### 3 Previous works

#### 3.1 Depth-first branch and bound algorithm for triangulation

To find the optimal triangulation, we investigate all possible elimination orders. Branch and bound is a general efficient algorithm for find-



**Figure 3:** Example of the vertex elimination and partially triangulated graphs induced by an elimination order that starts with sequence  $\{A, B\}$ . Left: Initial graph. Middle left: Partially triangulated graphs correspond to elimination order prefix  $\{A\}$ . Middle right: Partially triangulated graphs correspond to elimination order prefix  $\{A, B\}$ . Right: Final triangulated graph.

ing the optimal solution of a triangulation problem. To employ this algorithm, we must branch a search tree and compute a lower bound for each node.

First, we generate the search tree as follows. Each node corresponds to an elimination order prefix  $\tau$ . Each edge corresponds to a particular vertex being eliminated. Herein, we use the term "node" exclusively for a point in the search tree, and the term "vertex" exclusively for a point in the graph being triangulated. The exploration begins with *root node r* on the search tree, which corresponds to no vertex having been eliminated from G. The goal nodes corresponding to all vertices have been eliminated. Figure 2 depicts an example of a search tree.

To compute a lower bound for each node in the search tree, we must calculate the following with each node  $t$ :

- $t.H = (V, E \cup F)$ : Original graph with all fill-in edges F accumulated according to the elimination order prefix  $t.\tau$ .
- $t.R$ : Remaining vertices  $V \setminus W$ , where  $W$  are the all vertices of  $\tau$ .
- $t.C$ : A set of cliques for  $H$ .
- $t.tts$ : Total table size for the graph  $H$ , which is a lower bound for node  $t$ .

To compute  $t.tts$ , we must calculate all the cliques  $t.C$ . For this purpose, we can use a standard algorithm such as the well-known Bron-Kerbosch algorithm (BK algorithm) (Cazals and Karande, 2008). The following example

shows how we explore the leftmost path in Figure 2.

**Example 1.** Figure 3 depicts the vertex elimination process according to the leftmost path in Figure 2. The path follows the elimination order prefix  $\{A, B\}$ . The *root node*  $r$  corresponds to the graph on the left (initial graph), where no vertex has been eliminated. We can compute  $r.C = \{\{A, B, C\}, \{B, D\}, \{D, E\}, \{C, E\}\}$  using the BK algorithm. The TTS (assuming binary variables) is  $3 \cdot 2^2 + 2^3 = 20$ , which is a lower estimate of the TTS of the triangulated graph (Ottosen and Vomlel, 2010b).

The successor node  $t$  of  $r$  (corresponding to the elimination of vertex A) corresponds to the graph on the middle-left. The graph is the same as the initial one, so we can derive  $\text{tts}=20$ .

Then we explore successor note  $t'$  of  $t$  (corresponding to the elimination of vertex B). Therefore, the relevant graph corresponds to middle-right including the fill-in edge  $\{C, D\}$ . This process continues until the graph is triangulated. Finally, the cliques of the triangulated graph are  $C = \{\{A, B, C\}, \{B, C, D\}, \{C, D, E\}\}$ , and their TTS is  $3 \cdot 2^3 = 24$ .

We explained the search tree and how to compute a lower bound for each node. The branch and bound algorithm for optimal triangulation can be implemented in  $O(|V|)$  space and  $O(|V|!)$  time.

Ottosen and Vomlel (2010b) improved the naive branch and bound algorithm by adding the following pruning rules: (1) Graph reduction techniques called the simplicial vertex rule (Bodlaender et al., 2005), and (2) pruning based on a coalescing map, by using the fact that the remaining graph  $H[V \setminus W]$  is the same irrespective of the order the vertices in  $W$  have been eliminated (Dow and Korf, 2007). Although the Branch and Bound theoretically runs in  $O(|V|!)$ , it merely hits the upper bound. Ottosen and Vomlel (2010b) pointed out that their Branch and Bound with coalescing of nodes actually runs in  $O(2^{|V|})$  time.

Algorithm 1 shows Ottosen’s algorithm. The algorithm operates as follows: We initialize the best solution with the minimum fill-in heuris-

---

**Algorithm 1** Depth-first search with coalescing and upper-bound pruning.

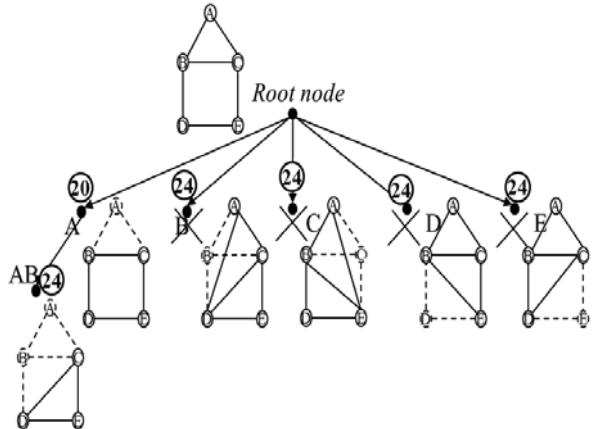
---

```

1: function TRIANGULATIONBYDFS(G)
2: Let s = (G, C(G), tts(G), V)
3: EliminateSimplicial(s)
4: if |V(s.R)|=0 then
5: return s
6: Let best=MinFill(s) \triangleright upper bound
7: map = \emptyset \triangleright Coalescing map
8: ExpandNode(s,best,map) return best
9: procedure EXPANDNODE(n,&best,&map)
10: for all v \in V(n.R) do
11: Let m = Copy(n)
12: EliminateVertex(m,v)
13: EliminateSimplicial(m)
14: if |V(m.R)|=0 then
15: if m.tts<best.tts then
16: Set best = m
17: else
18: if m.tts \geq best.tts then
19: continue
20: if map[m.R].tts \leq m.tts then
21: continue
22: Set map[m.R] = m
23: ExpandNode(m,best,map)

```

---



**Figure 4:** Example of the upper bound pruning. The number in circle is TTS of node. We first generate a sub-optimal solution according to the minimum fill-in heuristic, which is the elimination order beginning from A,B then use this upper bound 24 to prune the other nodes of which TTSs are more than or equal to this upper bound.

tic, set  $\text{best.tts}$  as the upper bound, and then start a branch and bound algorithm to seek a

**Algorithm 2** maintenance of cliques by local search

---

```

1: procedure UPDATE($G, \&C, tts, F$)
2: set $G' = (V, E \cup F)$
3: Let $U = \mathcal{V}(F)$
4: for all $C \in \mathcal{C}(G)$ do
5: if $C \cap U \neq \emptyset$ then
6: Set $tts = tts - ts(C)$
7: Set $\mathcal{C} = \mathcal{C} \setminus C$
8: let $C^{new} = \text{FindCliques}(G[\text{fa}(U, G')])$
9: for all $C \in C^{new}$ do
10: if $C \cap U \neq \emptyset$ then
11: Set $tts = tts + ts(C)$
12: Set $\mathcal{C} = \mathcal{C} \cup C$

```

---

better solution. When we find a node of which the lower bound of TTS is greater than best.tts, we prune the node. Figure 4 describes an example of the pruning procedure. If we find a better ordering, we update the best solution. The procedure EliminateVertex ( $\cdot$ ) simply eliminates a vertex from the remaining graph R and updates the cliques and TTS of the partially triangulated graph. The procedure EliminateSimplicial ( $\cdot$ ) removes all simplicial vertices from the remaining graph.

### 3.2 Dynamic clique maintenance

To compute  $t.\mathcal{C}$  for each node  $t$ , we can use a standard algorithm like the BK algorithm, which compute all cliques of a graph. Ottosen and Vomlel (2010b) pointed out that recomputing all cliques is unnecessary and proposed the following dynamic clique maintenance algorithm.

The general idea behind the algorithm is simple. Instead of searching for all 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 dynamic clique maintenance method is presented in Algorithm 2. As a side benefit, it also updates the TTS and the current graph. Algorithm 2 is derived based on the following theorem:

**Theorem 1** (Ottosen and Vomlel, 2010a). *Let  $G = (V, E)$  be an undirected graph, and let  $G'$*

**Algorithm 3** Clique maintenance with reduced search.

---

```

1: procedure UPDATE($G, \&C, tts, F, W$)
2: set $G' = (V, E \cup F)$
3: Let $U = \mathcal{V}(F)$
4: for all $C \in \mathcal{C}(G)$ do
5: if $C \cap U \neq \emptyset$ then
6: if $C \cap W = \emptyset$ then
7: Set $tts = tts - ts(C)$
8: Set $\mathcal{C} = \mathcal{C} \setminus C$
9: let $C^{new} = \text{FindCliques}(G[\text{fa}(U, G') \setminus W])$
10: for all $C \in C^{new}$ do
11: if $C \cap U \neq \emptyset$ then
12: Set $tts = tts + ts(C)$
13: Set $\mathcal{C} = \mathcal{C} \cup C$

```

---

$= (V, E \cup F)$  be the graph resulting from adding a set of new edges  $F$  to  $G$ . Let  $U = \mathcal{V}(F)$ . The  $\mathcal{C}(G')$  can be found by removing the cliques from  $\mathcal{C}(G)$  that intersect with  $U$  and adding cliques of  $G'[\text{fa}(U, G')]$  that intersect with  $U$ .

**Example 2.** Consider the middle-right graph in Figure 3. We update the graph  $G = (V, E)$  on the middle-left with the set of fill-in edges  $F = \{\{C, D\}\}$  (line 2 in Algorithm 2). The set  $U = \{C, D\}$  and  $\text{fa}(U, G') = \{A, B, C, D, E\}$ . We iterate through the existing cliques and remove those that intersect with  $U$  (lines 4–7), and then add all the cliques found in the subgraph  $G'[\text{fa}(U, G')]$  (lines 8–12) but which also intersect with  $U$ . We can observe that  $\mathcal{RC}(G, G') \cup \{A, B, C\}$  are removed and that  $\mathcal{NC}(G, G') \cup \{A, B, C\}$  are added. The clique  $\{A, B, C\}$  is removed and added again using this algorithm.

The example shows that the algorithm sometimes removes and adds the same cliques again. Although Ottosen's approach reduces the search range of the expensive enumeration algorithm  $\text{FindCliques}(\cdot)$  to a small subgraph, the method might have a potential performance problem such as duplicated cliques becoming very numerous. In the next section, we present a new algorithm that avoids this duplicated search.

## 4 Extended clique maintenance algorithm

In this section, we describe improvements of the clique maintenance algorithm that were made to perform clique enumeration algorithm on a smaller subgraph  $G[fa(U,G) \setminus W]$  than that of Ottosen and Vomlel (2010a). The clique enumeration algorithm is exponential with  $|V|$ . Therefore, this reduction of the search range is important. The proposed algorithm is described in Algorithm 3, where  $W$  is the set of vertices that have been eliminated.

The following example explains the algorithm:

**Example 3.** Consider again the middle-right graph in Figure 3. In our algorithm, the search range is limited to  $fa(U,G) \setminus W = \{B,C,D,E\}$ . In this example, we only add cliques  $\mathcal{NC}(G,G')$  and remove cliques  $\mathcal{RC}(G,G')$ .

Xiang and Lee (2006) describes a set of vertices called a crux which is central to their method for learning network structure. The algorithm described above may also be used to efficiently calculate the crux.

The correctness of this algorithm can be derived from the result below. Lemma 1 proves that our algorithm adds all new cliques  $\mathcal{NC}(G,G')$ . Lemma 2 proves that our algorithm removes all new cliques  $\mathcal{RC}(G,G')$ . Theorem 2 proves that our algorithm correctly updates the cliques for new graph.

**Lemma 1.** Let  $G$ ,  $G'$ ,  $F$ , and  $U$  be given as presented in Theorem 1.  $W$  are the vertices that have been eliminated. If  $C \in \mathcal{NC}(G,G')$ , then  $C \subseteq fa(U,G') \setminus W$ .

*Proof.* Let  $C$  be a new clique, it must include at least a new edge, i.e.  $C$  includes a vertex  $v \in U$ . Because  $C$  is complete, all vertices  $w \in C \setminus U$  must be connected to the vertex  $v$ , i.e.  $C \subseteq fa(U,G')$ .

Presuming that  $C \in \mathcal{C}(G')$ , which intersects with  $W$ , it must include a new edge, all vertices  $u \in W$  have been made simplicial by vertex elimination, which is a contradiction. Therefore, a new clique  $C \subseteq fa(U,G') \setminus W$ .  $\square$

**Lemma 2.** Let  $G$ ,  $G'$ ,  $F$ ,  $U$ , and  $W$  be given as

in Lemma 1. If  $C \in \mathcal{RC}(G,G')$ , then  $C$  intersects with  $U$  and disjoint with  $W$ .

*Proof.* Let  $C \in \mathcal{RC}(G,G')$ . Assuming that  $C \cap U = \emptyset$ , then for each  $v \in nb(C,G)$ ,  $C \not\subseteq nb(v,G')$  (otherwise  $C$  can not be a clique in  $G$ ). Therefore,  $C$  is still a clique in  $G'$ , which is a contradiction.

Let  $C \in \mathcal{RC}(G,G')$ . Under assumption  $C \cap W \neq \emptyset$ , for each  $w \in C \cap W$ , no  $v \in nb(C,G')$  exists such that  $nb(v,G')$  includes  $w$ . Therefore,  $C$  is still a clique in  $G'$ , which is a contradiction. Therefore, all remove clique  $C$  intersects with  $U$  and disjoint with  $W$ .  $\square$

**Theorem 2.** Let  $G$ ,  $G'$ ,  $F$ ,  $U$ , and  $W$  be given as presented in Lemma 1. The  $\mathcal{C}(G')$  can be found by removing the cliques from  $\mathcal{C}(G)$  that intersects with  $U$  and disjoint with  $W$ , and then by adding cliques of  $G'[fa(U,G') \setminus W]$  that intersect with  $U$ .

*Proof.* (1) We first show that all cliques in  $\mathcal{NC}(G,G')$  are added. From lemma 1, the subgraph  $G'[fa(U,G') \setminus W]$  includes all cliques in  $\mathcal{NC}(G,G')$ . Furthermore, any new clique  $C$  must intersect with  $U$  (otherwise it could not include a new edge). Therefore, we add all cliques in  $\mathcal{NC}(G,G')$ .

(2) We show that all cliques in  $\mathcal{RC}(G,G')$  are removed. From lemma 2, if  $C \in \mathcal{RC}(G,G')$ , then  $C \cap W = \emptyset$  and  $C \cap U \neq \emptyset$ . Therefore, we remove all potential cliques in  $\mathcal{RC}(G,G')$ .

(3) We consider that we can also remove a clique  $C \in \mathcal{C}(G') \cap \mathcal{RC}(G)$ . Because  $C$  intersects with  $U$  and disjoint with  $W$ , then  $C \subseteq fa(U,G') \setminus W$ , and  $C$  will be added again when we add cliques from  $G'[fa(U,G') \setminus W]$  that intersects with  $U$ .  $\square$

## 5 Search graph reduction

In this section, we exploit the following theorem to reduce the search tree:

**Theorem 3.** (Dirac, 1961) A triangulated graph with at least two nodes has at least two simplicial nodes.

Theorem 3 is applicable directly to depth-first search, as shown in algorithm 4.

**Algorithm 4** Depth-first search with pruning.

```

1: Insert lines 1–8 of Algorithm 1
2: procedure EXPANDNODE($n, \&best, \&map$)
3: Let $u = \text{SelectOneVertex}(n.R)$
4: for all $v \in \mathcal{V}(n.R) \setminus u$ do
5: Let $m = \text{Copy}(n)$
6: EliminateVertex(m, v)
7: EliminateSimplicial(m)
8: Insert lines 14–23 of Algorithm 1

```

**Table 1:** Results for random Bayesian networks.

| vertices | proposed<br>(second) | Ottosen's<br>(second) |
|----------|----------------------|-----------------------|
| 30       | 5.62                 | 12.08                 |
| 35       | 20.77                | 33.82                 |
| 40       | 46.39                | 97.3                  |
| 45       | 85.37                | 187.29                |
| 50       | 220.08               | 418.68                |
| 55       | 1256.17              | 2158.18               |
| 60       | 36324                | None                  |

Recalling that a node represents an elimination order prefix, then for an arbitrary vertex  $u$  which is to be eliminated next, according to Theorem 3, we can prune the corresponding node (line 4 in Algorithm 4). This pruning based on the fact: For any elimination order prefix  $\tau = \{v_1, v_2, \dots, v_{i-1}, t\}$ , there exists another  $\tau' = \{v_1, v_2, \dots, v_{i-1}, t'\}$  ( $t' \neq t$ ) can engender the same triangulation. As a result, the search tree size can be reduced from  $|V|!$  to  $|V-1|!$ . As Ottosen and Vomlel (2010b) pointed out, the Branch and Bound with coalescing of nodes actually runs in  $O(2^{|V|})$  time. Therefore, the actual runtime in this study is improved from  $O(2^{|V|})$  to  $O(2^{|V-1|})$ .

## 6 Experiments

In this section, we present experimental results obtained from running our algorithm on random Bayesian networks (Ide and Cozman, 2002) and on benchmark graphs from the Bayesian network repository. For comparison, we also solve each triangulation problem using Ottosen's algorithm. The algorithms were implemented using Java. All experiments were conducted on a Xeon-5675 3.0 GHz processor (Intel Corp.) with 12 GB of RAM.

**Table 2:** Results for Bayesian networks from the repository.

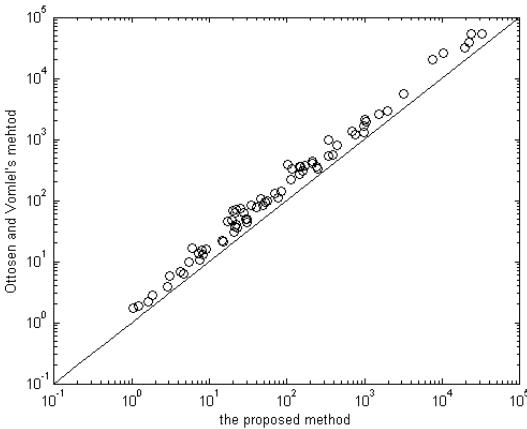
| BN         | vertices | proposed<br>(second) | Ottosen's<br>(second) |
|------------|----------|----------------------|-----------------------|
| Insurance  | 27       | 4.201                | 6.872                 |
| Water      | 32       | 21.696               | 65.682                |
| Mildew     | 35       | 19.915               | 67.433                |
| Alarm      | 37       | 0.006                | 0.012                 |
| Hailfinder | 56       | 21.871               | 70.004                |
| HEPAR2     | 70       | 0.054                | 0.115                 |
| WIN95PTS   | 76       | 158.259              | 299.181               |

All graphs that are described in Table 1 were generated using the BNGenerator system implemented by Ide and Cozman (2002). We generated 10 random graphs respectively for 30, 35, 40, 45, 50, 55 and 60 vertices. Each value in Table 1 shows the average time for the 10 graphs given the number of vertices, where “None” indicates that the algorithm obtained no solution in 24 hours. Additionally, we used seven well-known graphs in the Bayesian network repository. The results are presented in Table 2. The experiment results show that our algorithm is about 2 or 3 times faster than Ottosen et al.'s algorithm. The results also show that our algorithm can obtain the solution when  $|V| = 60$  and the Ottosen et al.'s algorithm can not obtain it in 24 hours.

Figure 5 shows the log–log plot of running time of proposed algorithm and that of Ottosen's algorithm for all random and repository networks, on which the x-axis is the our running time for triangulation, and the y-axis is Ottosen's running time. The values above the line indicate that our proposed method is superior to Ottosen's method. From Figure 5, our proposed algorithm is shown to be constantly faster than Ottosen's approach for both random Bayesian networks and repository networks.

## 7 Conclusions

The existing state-of-the-art algorithms for optimal triangulation are depth-first search and best-first search. Ottosen and Vomlel (2010b) pointed out that depth-first search is better than best-first search and proposed a depth-first search algorithm of which the complexity is  $O(\beta \cdot |V|!)$ , where  $\beta$  is the per-node overhead



**Figure 5:** Comparison of the running times of Ottosen’s method and the proposed method. X-axis shows results obtained using our method; the y-axis shows those obtained using Ottosen’s method. Values above the line indicate that our method is faster.

computation. We developed a new algorithm for the optimal triangulation of a Bayesian network. The proposed algorithm improves the computational complexity of the Ottosen’s algorithm in two ways.

For reduction of the computational cost  $\beta$  per node (section 4), we developed a new algorithm for maintaining the cliques of a dynamic graph. The new method is superior to Ottosen’s method because it searches cliques on a smaller local graph than Ottosen’s method does. This improves the computation cost from  $\beta$  to  $\gamma$ , where  $\gamma$  is strictly smaller than  $\beta$ .

For search tree shrinkage (section 5), we reduced the size of search tree from  $|V|!$  to  $(|V|-1)!$  using Dirac’s theorem.

The complexity of our proposed triangulation algorithm is  $O(\gamma \cdot (|V|-1)!)$ . Some numerical experiments demonstrated that the proposed method improves the traditional method.

For the future work, we will investigate other implementations of search graph reduction, e.g. the way proposed in (Bodlaender et al., 2006) may improve our algorithm in running time.

## References

- H. L. Bodlaender, A. M.C.A. Koster, and F. V. D. Eijkhof. 2005. Preprocessing rules for triangulation of probabilistic networks. *Computational Intelligence*, 21(3):286–305.

- H. L. Bodlaender, F. V. Fomin, A. M. C. A. Koster, D. Kratsch, and D. M. Thilikos. 2006. On exact algorithms for treewidth. In *Proceedings of the 14th conference on Annual European Symposium*, volume 14, pages 672–683.
- F. Cazals and C. Karande. 2008. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407:564 – 568.
- A. Darwiche. 2009. *Modeling and reasoning with Bayesian networks*. Cambridge University Press.
- G. A. Dirac. 1961. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25:71–76.
- P. A. Dow and R. E. Korf. 2007. Best-first search for treewidth. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, volume 2, pages 1146–1151. AAAI Press.
- V. Gogate and R. Dechter. 2004. A complete anytime algorithm for treewidth. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 201–208, Arlington, Virginia, United States. AUAI Press.
- J. S. Ide and F. G. Cozman. 2002. Random generation of bayesian networks. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, pages 366–375.
- F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. 1990. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282.
- T. J. Ottosen and J. Vomlel. 2010a. Honour thy neighbour—clique maintenance in dynamic graphs. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*, volume 2010-2. HIIT Publications.
- T. J. Ottosen and J. Vomlel. 2010b. All roads lead to rome—new search methods for optimal triangulation. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*, volume 2010-2. HIIT Publications.
- W. Wen. 1990. Optimal decomposition of belief networks. In *Proceedings of the Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 245–256.
- Y. Xiang and J. Lee. 2006. Learning decomposable markov networks in pseudo-independent domains with local evaluation. *Machine Learning*, 65:199–227.

# Decision-Theoretic Troubleshooting: Hardness of Approximation\*

Václav Lín

Faculty of Management, University of Economics, Prague

and

Institute of Information Theory and Automation of the ASCR

lin@utia.cas.cz

## Abstract

Troubleshooting is one of the application areas of Bayesian networks. Given a probabilistic model of a malfunctioning device, the task is to find the repair strategy with minimal expected cost. Except for simple cases, finding an optimal strategy is *NP*-hard. We show that optimal troubleshooting strategies are also hard to approximate.

## 1 Introduction

In decision-theoretic troubleshooting, we are given a probabilistic model of a man-made device. The model describes faults, repair actions and diagnostic actions addressing the faults. Knowing that the modeled device is in a faulty state, the task is to find the most cost-efficient strategy for fixing the device with available repair and diagnostic actions. This is a natural optimization problem that has been studied independently in various contexts since the early days of computing – the oldest works known to the author are (Johnson, 1956; Gluss, 1959).

Troubleshooting has become one of the application areas of Bayesian networks and as such it has received considerable attention over the last two decades. We refer to (Breese and Heckerman, 1996; Jensen et al., 2001; Ottosen, 2012) for discussion, references and survey of the most important results. The problem is known to be solvable in polynomial time under quite restrictive assumptions (to be discussed in Section 2). Otherwise, the problem is *NP*-hard (Vomlelová, 2003; Lín, 2011).

In Section 2, we recall several troubleshooting scenarios proposed in the literature. In Section 3 we identify a combinatorial optimization problem that captures their difficulty and show

that computing approximate solutions with performance guaranteed within certain bounds is *NP*-hard.

**Our Contribution.** We solve an open problem suggested by Ottosen (2012) – we show that troubleshooting with cost clusters forming an acyclic directed graph (see below) is both *NP*-complete and *NP*-hard to approximate.

We strengthen known *NP*-completeness results due to Vomlelová (2003) by showing hardness of approximation for troubleshooting scenarios containing either multiple dependend faults or dependent actions.

All the troubleshooting scenarios that we consider are closely related to a single combinatorial problem – *Min-sum Set Cover* (Feige et al., 2004). This can be used to develop novel troubleshooting algorithms in the future (see also Section 4).

## 2 Troubleshooting Models and Strategies

Bayesian networks for troubleshooting contain variables representing *faults*, repair actions, called simply *actions*, and diagnostic actions, called *questions*. We take several assumptions common in earlier literature (Jensen et al., 2001; Vomlelová and Vomlel, 2003):

Actions have only two possible outcomes – either the system is fixed after the action has been performed, or it remains in a faulty state.

---

\*This work was supported by the Institutional Research Support of the Faculty of Management, Prague University of Economics.

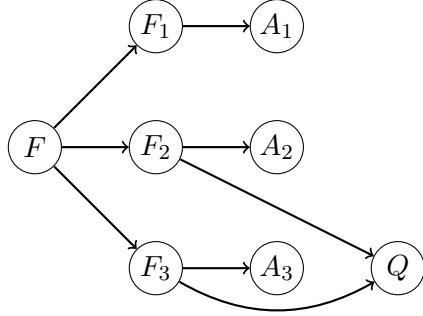


Figure 1: Bayesian network from Example 1. A troubleshooting model with single fault assumption and actions conditionally independent given the faults.

We assume that by performing the actions, we cannot introduce any new faults, and we know the outcome of any action immediately after its execution. Questions do not alter the state of the system, but may give useful information to direct the troubleshooting process. Each action or question has an associated cost. We take the assumption that these costs do not change over time.

Often, we take the *single fault assumption* – there can be but a single fault present in the system at any moment of time.

**Example 1.** A simple troubleshooting model is shown in Figure 1. There are three faults –  $F_1$ ,  $F_2$ ,  $F_3$ . To enforce the single fault assumption, we use a fault variable  $F$  with states  $\{1, 2, 3\}$  and define the probability tables for all  $F_i$  so that  $F_i = 1$  if and only if  $F = i$ . There are actions  $A_1$ ,  $A_2$ ,  $A_3$ , each addressing one of the faults. There is a question  $Q$  that can be used to discriminate between  $F_2$  and  $F_3$ .

Troubleshooting *strategy* is a policy governing the troubleshooting process. In general, it is a rooted directed tree with internal nodes labeled by actions and questions. Edges are labeled by outcomes of the actions and questions. The outdegree of nodes labeled by repair actions is exactly two, since we assume that each action has only two possible outcomes: “1” (system fixed) and “0” (system still in faulty state). An example of a troubleshooting strategy is in Figure 2. Let  $s$  be a troubleshooting strategy.

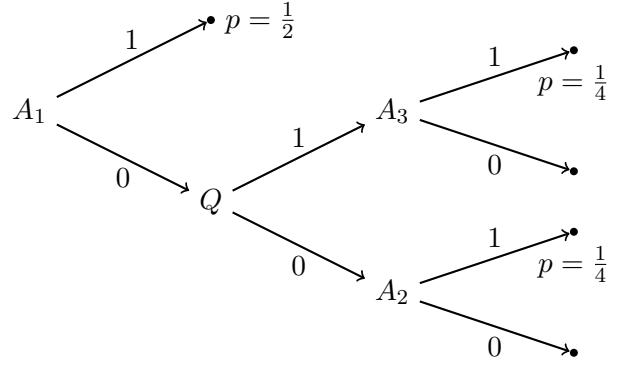


Figure 2: A troubleshooting strategy with actions  $A_1$ ,  $A_2$ ,  $A_3$  and a question  $Q$ . Nodes are labeled by actions and questions; edges are labeled by action or question outcomes. According to this strategy, each troubleshooting session will begin with action  $A_1$ . If it fails ( $A_1 = 0$ ), we use question  $Q$ . Depending on the outcome of  $Q$ , we either perform  $A_2$  or  $A_3$ . For further discussion, see Example 2.

Each path from the root of  $s$  to one of the leaves corresponds to a possible troubleshooting session starting in the root and terminating in the leaf. *Failure nodes* are all the terminal nodes  $l$  of  $s$  for which the corresponding troubleshooting session fails to fix the system. For a strategy  $s$ , we use this notation:

- $\mathcal{L}(s)$  – the set of terminal nodes,
- $t(l)$  – the cost of performing all the actions and questions on the path from the root of  $s$  to a terminal node  $l \in \mathcal{L}(s)$ ,
- $\mathcal{P}(l)$  – the probability of reaching node  $l$ ,
- $\mathcal{L}^-(s) \subset \mathcal{L}(s)$  – the set of failure nodes,
- $c_P$  – the penalty for not fixing the system.

The goal is to construct a strategy  $s$  minimizing the *expected cost of repair*

$$ECR(s) = \sum_{l \in \mathcal{L}(s)} \mathcal{P}(l) \cdot t(l) + \sum_{l \in \mathcal{L}^-(s)} \mathcal{P}(l) \cdot c_P . \quad (1)$$

Heuristic search algorithms for computing optimal troubleshooting strategies are described in (Vomlelová and Vomlel, 2003).

**Example 2.** We continue with Example 1, by showing how to evaluate the *ECR* of the strategy shown in Figure 2. Assume that the probabilities specified for the Bayesian network shown in Figure 1 are:

- $\mathcal{P}(F = 1) = \frac{1}{2}$ ,
- $\mathcal{P}(F = 2) = \mathcal{P}(F = 3) = \frac{1}{4}$ ,
- for  $i = 1, 2, 3$ ,  $F_i = 1$  if and only if  $F = i$ ,
- for  $i = 1, 2, 3$ ,  $A_i = 1$  if and only if  $F_i = 1$ ,
- $Q = 1$  if and only if  $F_3 = 1$ .

Failure terminal nodes of the strategy have zero probability, therefore we can disregard the penalty term  $c_P$  of Formula 1. Assume the cost of  $Q$  and all  $A_i$ 's is one. Summing over the terminal nodes with positive probability (see the  $p$ 's in Figure 2), we get

$$ECR = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 = \frac{3}{2}.$$

### Troubleshooting without Questions.

When there are no questions, the troubleshooting strategy is just a sequence of actions. The actions are performed in order and the troubleshooting session continues until the fault is fixed or all the actions have been used.<sup>1</sup> When we assume that all the actions of a sequence  $A_1, \dots, A_n$  are sufficient to fix the fault, we can ignore the penalty term of Formula 1. Then we can use formula (2) to compute the *ECR*:

$$\sum_{i=1}^n \mathcal{P}\left(\bigcup_{j < i} \{A_j = 0\} \cup \{A_i = 1\}\right) \cdot \sum_{j \leq i} c(A_j), \quad (2)$$

i.e., we multiply the cost of first  $i$  actions by the probability of fixing the fault with the  $i$ -th action. Finding optimal strategies in polynomial time is known to be possible under quite restrictive assumptions (Jensen et al., 2001):

<sup>1</sup>Note that this is a simplification. We could consider models where it is possible to terminate the troubleshooting process *before* we have tried all the available actions. Instead of continued troubleshooting, we would pay some fixed penalty. In the real life, this happens – we quite often decide to buy a new device before we have tried everything possible to fix an old one.

- There is exactly one fault present.
- The actions are conditionally independent given the faults and each action addresses exactly one fault.
- The action costs are constant over time and there are no questions.

Scenarios conforming to these assumptions are called *basic troubleshooting*. Optimal troubleshooting sequence is computed by ordering the actions so that the ratios  $P(A_j = 1)/c(A_j)$  are non-increasing.<sup>2</sup>

**Cost Clusters.** When troubleshooting a large piece of machinery (such as the car engine), it is often necessary to disassemble the machine to perform certain actions. In general, some of the troubleshooting actions may require common initialization or preparatory work. To model such situations, Langseth and Jensen (2001) proposed an extension of the basic troubleshooting model, where the set of actions is partitioned into disjoint subsets, called *cost clusters*. To access actions within a cluster  $\mathcal{K}_i$ , we have to pay additional cost,  $c(\mathcal{K}_i)$ . Once  $c(\mathcal{K}_i)$  is paid, we can use actions from  $\mathcal{K}_i$  at any time, possibly mixed with actions from other clusters. Ottosen and Jensen (2010) have generalized the cost cluster scenario by allowing the cost clusters to form a tree and gave a polynomial-time algorithm for finding optimal troubleshooting sequences. Ottosen (2012) suggested a further generalization of the problem, where the cost clusters are allowed to form an acyclic directed graph. A simple model with cost cluster graph is shown in Figure 3.

Note that the cost cluster scenarios discussed so far differ from the scenario where we have cost clusters “*without* inside information” (Langseth and Jensen, 2001). Without inside information, we have to close the most recently open cluster whenever we need to check the outcome of troubleshooting actions, and the cost

<sup>2</sup>In different contexts, this observation has already been made in the 1950's – by Bellman (1957) in Dynamic Programming and by Smith (1956) in Scheduling.

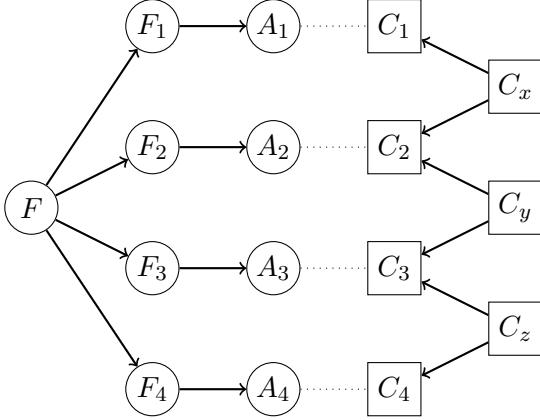


Figure 3: Troubleshooting with cost clusters. At the left side is Bayesian network with faults  $F_i$  and actions  $A_i$ . At the right side is the graph of cost clusters. To access, say, action  $A_2$ , we have to open clusters  $C_y$  and  $C_2$  (or  $C_x$  and  $C_2$ ).

$c(\mathcal{K}_i)$  is paid whenever we open  $\mathcal{K}_i$  again. Troubleshooting cost clusters without inside information is *NP-hard* (Lín, 2011).

**Modifications of the Cost Cluster Model.** Gluss (1959) described a problem which is similar to the troubleshooting scenarios discussed in previous paragraph. Its complexity is unknown. Assume that there are independent faults and each of them is addressed by single action. Any number of faults can occur at the same time. The set of actions is partitioned into disjoint cost clusters (“modules” in (Gluss, 1959)). When we decide to perform an action from a particular cluster  $\mathcal{K}_i$ , we pay the cluster opening cost  $c(\mathcal{K}_i)$ . If we continue performing actions from  $\mathcal{K}_i$ , no additional cost is incurred. However, a cluster opening cost is incurred *whenever* we switch between clusters. Outcome of an action is known immediately after it is performed (contrary to the cost cluster scenario without inside information).

When we take the single fault assumption, we can use observations made in (Lín, 2011) and show that this troubleshooting scenario is equivalent to a scheduling problem called “scheduling job families with sequence-independent setup times on a single machine to minimize total weighted completion

time”, denoted  $1|s_f| \sum w_j C_j$  in the Scheduling notation. This problem is mentioned by Potts and Kovalyov (2000). They give references to branch-and-bound algorithms for the problem but report that its complexity status is unresolved. To our knowledge, this is still true today.

### 3 Hardness of Approximation

We assume the reader is familiar with basic concepts of computational complexity theory. Therefore we only summarize some basic terminology used later on. For an introduction to the theory, we refer to textbooks such as (Garey and Johnson, 1979; Arora and Barak, 2009).

Given a minimization problem  $L$ , constant  $\rho > 1$ , and a polynomial-time algorithm  $A$  for  $L$ , we say that  $A$  is a *polynomial  $\rho$ -approximation algorithm* if for all instances  $x$  of problem  $L$  we have  $\rho \geq A(x)/\text{opt}(x)$ . By  $A(x)$  we denote the objective value returned by algorithm  $A$  when applied to instance  $x$ , and by  $\text{opt}(x)$  we denote the optimum of  $x$ . We say that it is *NP-hard* to approximate  $L$  within ratio  $\rho$ , if there is a polynomial-time reduction  $\phi$  from *3SAT* to  $L$ , such that  $\phi$  combined with a hypothetical  $\rho$ -approximation algorithm for  $L$  would make it possible to decide *3SAT* in polynomial time. By *NP*-completeness of *3SAT*, this would imply  $P=NP$ . Inapproximability results are now known for many problems (Johnson, 2006).

#### 3.1 Reductions

In this section, we treat troubleshooting scenarios without questions. We isolate several features of troubleshooting scenarios and show that each of them is sufficient to make troubleshooting hard to approximate. These features are: multiple dependent faults, dependent actions, acyclic directed graph of cost clusters.

The complexity of the troubleshooting scenarios studied in this paper is captured by single combinatorial problem that combines covering and sequencing – *Min-sum Set Cover*.

**Definition 1** (Min-sum Set Cover (*MSSC*)). Input: A finite set  $U$ , a collection of subsets  $\mathcal{C} = \{S \subseteq U\}$ .

Objective: Find a linear ordering  $\pi$  of  $\mathcal{C}$  minimizing the function

$$\sigma(U, \mathcal{C}, \pi) = \sum_{u \in U} \pi(u),$$

where  $\pi(u)$  is the index of the first set  $S \in \mathcal{C}$  covering  $u$  under the ordering  $\pi$ .

**Theorem 1** (Feige et al. (2004)). *MSSC has no polynomial  $(4 - \epsilon)$ -approximation algorithm for any  $\epsilon > 0$  unless P=NP.*

*Remark.* In the proofs of Theorems 2, 3, 4, we will assume that any *MSCC* instance  $(U, \mathcal{C})$  used in the reductions is such that  $U = \bigcup\{S \in \mathcal{C}\}$ , i.e., the set cover exists. We can make such an assumption without a loss of generality, since the hardness-of-approximation proof in (Feige et al., 2004) works with set systems for which set covers exist.

For later use, we record two simple lemmas. The first one follows from the definition. The second lemma is a consequence of the first one.

**Lemma 1.** *Consider an *MSCC* instance  $(U, \mathcal{C})$  and an ordering  $\pi$  of  $\mathcal{C}$ . Denote by  $U_{\pi(i)}$  the set of  $u \in U$  first covered by the  $i$ -th set of  $\pi$ . Then*

$$\sigma(U, \mathcal{C}, \pi) = \sum_{i=1}^{|\mathcal{C}|} |U_{\pi(i)}| \cdot i. \quad (3)$$

**Lemma 2.** *When  $U = \bigcup\{S \in \mathcal{C}\}$  and  $\pi^*$  is an optimal ordering of  $\mathcal{C}$ , then*

$$|U| \leq \sigma(U, \mathcal{C}, \pi^*) \leq \sum_{i=1}^{|U|} i. \quad (4)$$

We will reduce *MSSC* to several troubleshooting scenarios. In this section, we will work with scenarios without questions and will use Formula 2 for computation of the *ECR*. Let  $\pi$  be a linear ordering of actions  $\{A_i\}_{i=1}^n$  and define

$$p_{\pi(i)} = \mathcal{P}(\bigcup_{j < i} \{A_{\pi(j)} = 0\} \cup \{A_{\pi(i)} = 1\}).$$

Formula 2 then becomes

$$ECR = \sum_{i=1}^n p_{\pi(i)} \cdot \sum_{j \leq i} c(A_{\pi(j)}). \quad (5)$$

In definitions 2, 3, 4 we formally define simple troubleshooting scenarios, each of them isolating one property:

- single fault and dependent actions (Definition 2),
- multiple dependent faults (Definition 3),
- single fault, independent actions and cost clusters forming a DAG (Definition 4).

We show that already these very simple scenarios are hard to approximate.

In the proofs to follow, all the sets, collections of variables etc. are finite. All random variables are discrete.

**Definition 2** (Troubleshooting with Dependent Actions (*TSDA*)). Input: A random variable  $F$  with values (*faults*)  $f_1, \dots, f_n$ , a probability distribution  $\mathcal{P}(F)$ , a set of *actions*  $\{A_i\}_{i=1}^k$ . Each action fixes a subset of faults  $F(A_i) \subseteq \{f_j\}_{j=1}^n$  with certainty and no other faults. Each fault is fixed by at least one action.

Objective: Find a linear ordering of actions minimizing the expected cost of repair.

**Theorem 2.** *TSDA has no polynomial  $(4 - \epsilon)$ -approximation algorithm for any  $\epsilon > 0$  unless P=NP.*

*Proof.* We show that a special case of Troubleshooting with Dependent Actions is equivalent to Min-sum Set Cover. We reduce an *MSCC* instance  $(U, \mathcal{C})$  to a troubleshooting problem as follows. Create a fault variable  $F$  with a set of values  $\{f_u\}_{u \in U}$ . Distribution  $\mathcal{P}(F)$  is uniform. For each  $S \in \mathcal{C}$  create an action  $A_S$  with cost one.  $A_S$  fixes  $f_u$  with probability one if and only if  $f_u \in S$ . Let  $k = |\mathcal{C}|$ . Given an arbitrary linear ordering  $A_{\pi(1)}, \dots, A_{\pi(k)}$  of the actions, denote by  $F_{\pi(i)}$  the set of faults first covered by the  $i$ -th action of the ordering. The expected cost of repair (see Formula 5) is

$$\sum_{i=1}^k p_{\pi(i)} \cdot i = \sum_{i=1}^k i \cdot |F_{\pi(i)}| / |U|.$$

Using Lemma 1, we see that the *ECR* multiplied by  $|U|$  is equal to  $\sigma(U, \mathcal{C}, \pi)$ . Therefore any approximation algorithm for *TSDA* could be used to approximate  $(U, \mathcal{C})$  with the same approximation ratio.  $\square$

**Definition 3** (Troubleshooting with Dependent Faults (*TSDF*)). Input: A Bayesian network representing probability distribution  $\mathcal{P}(\mathbf{F})$ , binary random variables  $F_1, \dots, F_n \in \mathbf{F}$  (*faults*), a set of *actions*  $\{A_i\}_{i=1}^n$ . For each fault, there is exactly one action that fixes it with certainty. Objective: Find a linear ordering  $\pi$  of actions that minimizes the *ECR*.

**Theorem 3.** *TSDF has no polynomial  $(4 - \epsilon)$ -approximation algorithm for any  $\epsilon > 0$  unless P=NP.*

*Proof.* We use an idea by Vomlelová (2003) to transform the *TSDA* instance constructed in proof of Theorem 2 to *TSDF*. First, construct a Bayesian network for the *TSDA* with vertex set

$$\{F\} \bigcup \{F_u\}_{u \in U} \bigcup \{A_S\}_{S \in \mathcal{C}}$$

and edge set

$$\{F \rightarrow F_u\}_{u \in U} \bigcup \{F_u \rightarrow A_S\}_{u \in S}.$$

Variable  $F$  has states  $\{f_u\}_{u \in U}$  and an uniform probability distribution. All the other variables have deterministic distributions of probability:  $F_u = 1$  if and only if  $F = f_u$ , and  $A_S = 1$  if and only if at least one parent  $F_u$  of  $A_S$  has value 1. An example of such a network is shown in the left side of Figure 4 (for now, ignore the dotted part). We use the network to create a *TSDF* problem. Add to the network  $k$  new vertices  $A'_S$  and edges  $A_S \rightarrow A'_S$  (see the dotted part of Figure 4). The new vertices are *actions* of the *TSDF* instance, the original actions  $A_S$  are now *faults*. Cost of the new actions is one. The optimal *ECR* of the original *TSDA* problem is the same as the optimal *ECR* of the constructed *TSDF* problem.  $\square$

*Remark.* In Definition 3, we place no restriction on  $\mathcal{P}(\mathbf{F})$  except that it is represented by a Bayesian network. Bayesian network inference is in general a hard problem in itself (see e.g. Kwisthout et al. (2010) for references). In our proof of Theorem 3 we have constructed a Bayesian network for which the inference is easy and yet, the Troubleshooting problem is hard.

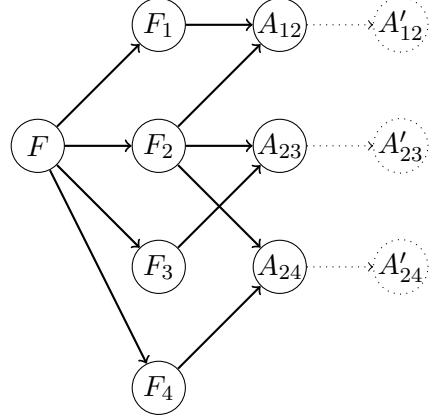


Figure 4: Model with dependent actions for  $U = \{1, 2, 3, 4\}$  and  $\mathcal{C} = \{\{1, 2\}, \{2, 3\}, \{2, 4\}\}$ . The dotted edges and vertices show extension to a model with dependent faults.

**Definition 4** (Troubleshooting with DAG Cost Clusters (*TSCC*)). Input: A random variable  $F$  with values (*faults*)  $f_1, \dots, f_n$ , a probability distribution  $\mathcal{P}(F)$ , a set of *actions*  $\{A_i\}_{i=1}^n$ . For each fault, there is exactly one action that fixes it with certainty. There is an acyclic directed graph  $(V, E)$  of cost clusters. Vertices  $v \in V$  represent cost clusters and edges  $u \rightarrow v$  show that cluster  $v$  can be accessed from  $u$ . Each action  $A_i$  is assigned to some cluster  $v \in V$ , and each cluster contains *zero or more* actions. The cost of opening cluster  $v$  is  $c(v) \geq 0$ .

Objective: Find a schedule of cluster opening and troubleshooting actions minimizing the expected cost of repair.

**Theorem 4.** *TSCC has no polynomial  $(3 - \epsilon)$ -approximation algorithm for any  $\epsilon > 0$  unless P=NP. The result holds even when the cost cluster graph is bipartite.*

*Proof.* We again reduce Min-sum Set Cover. For each item  $u \in U$ , create a fault  $f_u$  with probability  $P(F = f_u) = 1/|U|$  and an action  $A_u$ , solving exclusively  $f_u$ . Each action is contained in an associated cost cluster,  $C_u$ . The cost of opening  $C_u$  and performing  $A_u$  is one. For each set  $S \in \mathcal{C}$ , create an empty cluster  $C_S$  with cost  $c$  ( $c$  is a positive constant to be discussed later). Create directed edges from the

$C_S$ 's to the  $C_u$ 's according to set membership – there is edge  $C_S \rightarrow C_u$  if and only if  $S \ni u$ . An example of a cost cluster model created this way is in Figure 3. In any optimal schedule, each cluster  $C_u$  is opened right before performing  $A_u$ ; therefore, we shall not mention opening of the  $C_u$ 's in the rest of the proof. Any troubleshooting sequence has to contain all the actions  $A_u$ . Their order is arbitrary, since their costs and probabilities of success are uniform. Thus the  $ECR$  can be decomposed as a sum of the expected cost of actions and the expected cost of opening “top-level” clusters  $C_S$ . Assume the clusters  $\{C_S\}_{S \in \mathcal{C}}$  are indexed by integers  $1, \dots, k$  and are opened in sequence  $C_1, \dots, C_k$ . Using Formula 5, we get

$$\underbrace{\sum_{i=1}^n \frac{i}{n}}_{\text{actions}} + \underbrace{\sum_{j=1}^k P(C_j) \cdot j c}_{\text{top-level clusters}} = \frac{n+1}{2} + c \cdot \sum_{j=1}^k P(C_j) \cdot j,$$

where  $n = |U|$ ,  $k = |\mathcal{C}|$  and  $P(C_j)$  is the probability that by opening  $C_j$  we make accessible an action that fixes the fault and therefore  $C_j$  is the last cluster of the sequence that needs to be open. We assume that once a cluster  $C_S$  is open, we perform all the actions accessible from  $C_S$  except for those that have already been performed. Indeed, if we did not perform the actions greedily after opening each cost cluster, the  $ECR$  would increase, because some of the cost clusters could be open needlessly. With this assumption,  $P(C_j) = |F_{\pi(j)}|/n$ , where  $F_{\pi(j)}$  is the set of actions first made available by opening the  $j$ -th cluster. Let  $\pi$  be some ordering of  $\mathcal{C}$  and let  $\sigma(U, \mathcal{C}, \pi)$  be the value given by Formula 3. Then there is a corresponding troubleshooting sequence specified by ordering  $\pi$  with

$$ECR(\pi) = \frac{n+1}{2} + \frac{c}{n} \cdot \sigma(U, \mathcal{C}, \pi), \quad (6)$$

and the correspondence of *MSSC* solutions and *TSCC* solutions is one to one.

We conclude the proof by showing that for certain value of  $c$ ,

$$\frac{ECR(\pi)}{ECR(\pi^*)} < 3 \text{ implies } \frac{\sigma(U, \mathcal{C}, \pi)}{\sigma(U, \mathcal{C}, \pi^*)} < 4. \quad (7)$$

By Theorem 1, this would imply  $P=NP$ . Using Formula 6, we can write inequality

$$\frac{\sigma(U, \mathcal{C}, \pi)}{\sigma(U, \mathcal{C}, \pi^*)} = \frac{ECR(\pi) - (n+1)/2}{ECR(\pi^*) - (n+1)/2} < 4 \quad (8)$$

which is equivalent to

$$\frac{ECR(\pi)}{ECR(\pi^*)} < 4 - \frac{\frac{3}{2}(n+1)}{ECR(\pi^*)} \quad (9)$$

Now, for any lower bound  $\underline{ECR}(\pi^*)$  of  $ECR(\pi^*)$ , the inequality

$$\frac{ECR(\pi)}{\underline{ECR}(\pi^*)} < 4 - \frac{\frac{3}{2}(n+1)}{\underline{ECR}(\pi^*)} \quad (10)$$

implies inequality 9. To get the estimate  $\underline{ECR}$ , we use Formula 6 and Lemma 2:

$$\underline{ECR}(\pi^*) = \frac{n+1}{2} + \frac{c}{n} \cdot n = \frac{n+1}{2} + c.$$

When we set  $c = n+1$  and use  $\underline{ECR}$  in inequality 10, we finally prove implication 7 via inequalities 9 and 8.  $\square$

*Remark.* Note that had we used a greater value for the constant  $c$  in the proof, we would get an approximation ratio closer to 4, yielding a stronger theorem.

**Corollary 1.** *Troubleshooting with DAG Cost Clusters is NP-complete.*

*Proof.* NP-completeness is a concept defined for decision problems. Therefore we have to consider the *decision variant* of *TSCC*: given an arbitrary positive constant  $K$  and an instance  $x$  of *TSCC*, is it true that  $opt(x) \leq K$ ? This decision problem clearly belongs to class *NP* – once we guess the cluster opening / action schedule  $s(x)$ , it is easy to compute the  $ECR$  in polynomial time and check  $ECR(s(x)) \leq K$ . *NP*-hardness is implied by Theorem 4.  $\square$

## 4 Summary and Future Work

In the proofs, we have seen that the three troubleshooting scenarios are closely related to a well studied combinatorial problem *Min-sum Set Cover*. The natural next step is to have

a look at the algorithms developed for the mentioned problem and see how well do they apply to troubleshooting. In particular, a simple greedy algorithm for *MSSC* achieves approximation ratio 4, and hence the bound given by Theorem 1 is tight (Feige et al., 2004).

An open problem is the hardness of approximation of troubleshooting with questions. The problem has been shown to be *NP*-hard by Vomlelová (2003). The *NP*-hardness can be shown by a simple reduction from the *Decision Tree Problem* (Garey and Johnson, 1979), which has recently been shown to be hard to approximate by Adler and Heeringa (2012). However, it is not obvious how to extend the reduction to prove also hardness of approximation for troubleshooting with questions.

The Troubleshooting community grew in the 1990's without knowing about the very early works such as (Johnson, 1956) and (Gluss, 1959). It is worthwhile to perform further bibliographic research to see whether their work has been carried on in some communities other than our own.

## Acknowledgments

I thank Jirka Vomlel for useful discussions.

## References

- Micah Adler and Brent Heeringa. 2012. Approximating optimal binary decision trees. *Algorithmica*, 62(3-4):1112–1121.
- Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press.
- Richard E. Bellman. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.
- John S. Breese and David Heckerman. 1996. Decision-theoretic Troubleshooting: A framework for repair and experiment. In Eric Horvitz and Finn Verner Jensen, editors, *UAI*, pages 124–132. Morgan Kaufmann.
- Uriel Feige, László Lovász, and Prasad Tetali. 2004. Approximating Min Sum Set Cover. *Algorithmica*, 40(4):219–234.
- Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.
- Brian Gluss. 1959. An optimum policy for detecting a fault in a complex system. *Operations Research*, 7(4):468–477.
- Finn Verner Jensen, Uffe Kjærulff, Brian Kristiansen, Helge Langseth, Claus Skaanning, Jiří Vomlel, and Marta Vomlelová. 2001. The SACSO methodology for troubleshooting complex systems. *AI EDAM*, 15(4):321–333.
- Selmer M. Johnson. 1956. *Optimal Sequential Testing*. Number RM1652. Rand Corporation.
- David S. Johnson. 2006. The NP-completeness column: The many limits on approximation. *ACM Transactions on Algorithms*, 2(3):473–489.
- Johan Kwisthout, Hans L. Bodlaender, and Linda C. van der Gaag. 2010. The necessity of bounded treewidth for efficient inference in Bayesian networks. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 237–242. IOS Press.
- Helge Langseth and Finn Verner Jensen. 2001. Heuristics for two extensions of basic troubleshooting. In Henrik Hautop Lund, Brian H. Mayoh, and John W. Perram, editors, *SCAI*, volume 66 of *Frontiers in Artificial Intelligence and Applications*, pages 80–89. IOS Press.
- Václav Lín. 2011. Extensions of decision-theoretic Troubleshooting: Cost clusters and precedence constraints. In Weiru Liu, editor, *ECSQARU*, volume 6717 of *Lecture Notes in Computer Science*, pages 206–216. Springer.
- Thorsten J. Ottosen and Finn Verner Jensen. 2010. The cost of troubleshooting cost clusters with inside information. In Peter Grünwald and Peter Spirtes, editors, *UAI*, pages 409–416. AUAI Press.
- Thorsten J. Ottosen. 2012. *Solutions and Heuristics for Troubleshooting with Dependent Actions and Conditional Costs*. Aalborg University, Denmark.
- Chris N. Potts and Mikhail Y. Kovalyov. 2000. Scheduling with batching: A review. *European Journal of Operational Research*, 120(2):228–249.
- Wayne E. Smith. 1956. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66.
- Marta Vomlelová and Jiří Vomlel. 2003. Troubleshooting: NP-hardness and solution methods. *Soft Comput.*, 7(5):357–368.
- Marta Vomlelová. 2003. Complexity of Decision-theoretic Troubleshooting. *Int. J. Intell. Syst.*, 18(2):267–277.

# A Novel LTM-based Method for Multi-partition Clustering

Tengfei Liu, Nevin L. Zhang, Kin Man Poon, Hua Liu  
The Hong Kong University of Science and Technology  
`{liutf, lzhang, lkmpoon, aprillh}@cse.ust.hk`

Yi Wang  
National University of Singapore  
`wangy@comp.nus.edu.sg`

## Abstract

Early research work on clustering usually assumed that there was one true clustering of data. However, complex data are typically multifaceted and can be meaningfully clustered in many different ways. There is a growing interest in methods that produce multiple partitions of data. One such method is based on latent tree models (LTM). But previous methods for learning general LTM are computationally inefficient. In this paper, we propose a fast algorithm for learning LTM. Empirical results on two real world datasets are given to show that our method can produce rich and meaningful partitions.

## 1 Introduction

There are several clustering methods that produce multiple partitions. We refer to them as *multi-partition clustering (MPC)* methods. MPC methods, according to the way that partitions are found, can be divided into two categories: *sequential MPC* methods and *simultaneous MPC* methods.

*Sequential MPC* methods produce multiple partitions sequentially. One such kind of method is known as *alternative clustering* (Cui *et al.*, 2007; Gondek and Hofmann, 2007; Qi and Davidson, 2009). It aims to discover a new clustering that is different from a previously known clustering. The key issue is how to ensure the novelty of the new clustering. One can repeatedly apply such methods to produce a sequence of clusterings.

*Simultaneous MPC* methods, on the other hand, produce multiple partitions simultaneously. Both distance-based and model-based methods have been proposed. The distance-based methods (Jain *et al.*, 2008; Niu *et al.*, 2010) require as inputs the number of partitions and the number of clusters in each partition. They try to optimize the quality of each

individual partition while keeping different partitions as dissimilar as possible. Model-based methods fit data with a probabilistic model that contains multiple latent variables. Each latent variable represents a soft partition. Unlike distance-based methods, model-based methods can automatically determine the number of partitions and the number of clusters in each partition based on statistical principles.

Among the model-based methods, Galimberti and Soffritti (2007) and Guan *et al.* (2010) assume that each latent variable, which gives one view of data, is associated with a subset of attributes. The subsets for different latent variables are disjoint. A latent variable is independent of all the other latent variables and all the attributes that are not associated with it. On the other hand, Zhang (2004) and Poon *et al.* (2010) use latent tree models (LTM). Different from aforementioned methods, the latent variables in LTM are connected and form an interpretable tree structure.

This paper is concerned with the use of LTM for producing multiple partitions of categorical data. Currently, there is a lack of efficient algorithms for learning general LTM. In this paper we propose a fast algorithm to learn LTM. Em-

pirical results are also given to show that the new algorithm can produce rich and meaningful clustering results.

## 2 Latent Tree Model

Technically an LTM is a Markov random field over an undirected graph, where variables at leaf nodes are observed and variables at internal nodes are hidden. For technical convenience, we often root an LTM at one of its latent nodes and regard it as a directed graphical model, i.e., a Bayesian network. An example of LTM is shown in Figure 1. In the example model, the Y-variables are latent variables. The number in parenthesis is called cardinality which indicates the number of states of the latent variable. The leaf nodes in this example are different words which take binary values to indicate presence or absence of the word.

Throughout the paper, we use the term ‘node’ interchangeably with ‘variable’, and term ‘leaf node’ interchangeably with ‘attribute’. A set of attributes that are connected to the same latent variable is called a *sibling cluster*. Attributes in the cluster are said to be *siblings*. For example, in Figure 1, attributes *austin*, *utexas*, *texas* and *ut* form one sibling cluster because they are all connected to latent node  $Y_1$ .

The numerical information of LTM includes a marginal distribution  $P(Y_1)$  for the root  $Y_1$  and one conditional distribution for each edge. For example, for the edge  $Y_2 \rightarrow object$ , we have distribution  $P(object | Y_2)$ . The product of these distributions defines a joint distribution over all the latent and observed variables.

To learn an LTM from a dataset  $\mathbf{D}$ , one needs to determine: (1) the number of latent variables, (2) the cardinality of each latent variable, (3) the connections among the latent and observed variables, and (4) the probability parameters. We use  $m$  to denote the information for the first three items and  $\theta$  to denote the collection of parameter values. We aim at finding the pair  $(m, \theta^*)$  where  $\theta^*$  is the maximum likelihood estimate of the parameters and  $m$  maximizes the BIC score (Schwarz, 1978):

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

where  $d(m)$  is the number of free parameters in  $m$  and  $N$  is the sample size.

Several algorithms for learning LTM have been proposed. The latest algorithm for learning general LTM is the one called EAST (Chen *et al.*, 2012, 2008). It is a search-based method and is capable of producing good models for data sets with dozens of attributes. However, it is not efficient enough for data sets with more than 100 attributes. There are two other algorithms that are more efficient than EAST, but they focus on special LTM. Harmeling and Williams (2010) consider only binary trees, while Choi *et al.* (2011) assume all the variables share the same domain. Neither methods are intended for cluster analysis.

## 3 The Bridged Islands Algorithm

We now set out to present a new algorithm for learning general LTM that is more efficient than previous algorithm EAST. The new algorithm proceeds in four steps:

1. Partition the set of attributes into sibling clusters;
2. For each sibling cluster introduce a latent variable and determine the number of states of this variable;
3. Determine the connections among the latent variables so that they form a tree;
4. Refine the model.

If we imagine the sibling clusters formed in Step 1, together with the latent variables added in Step 2, as islands in an ocean, then the islands are connected in Step 3. So we call the algorithm the *bridged islands (BI)* algorithm.

### 3.1 Determining Sibling Clusters

The sibling cluster determination is the first step of the new algorithm. It is based on two intuitions: First, in an LTM, attributes from the same sibling cluster tend to be more closely correlated than those from different sibling clusters; Second, if two attributes are siblings in the optimal LTM for one set of attributes, they should also be siblings in the optimal LTM for a subset of the attributes. We determine the

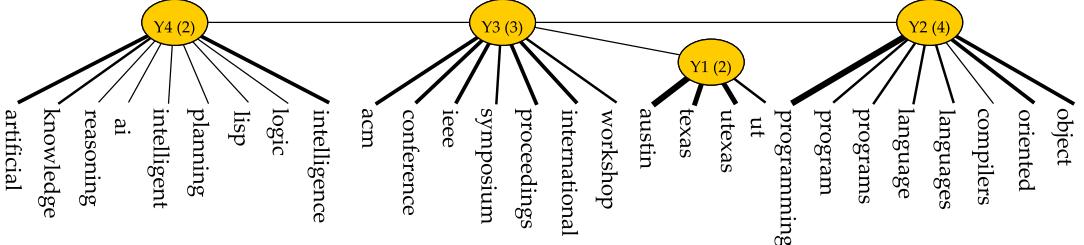


Figure 1: An example of latent tree model. The width of edges represent strength of probabilistic dependence.

first sibling cluster as follows. There is a working subset of attributes. Initially, it contains the pair of attributes with the highest *mutual information* (MI). Here MI is computed from the empirical distribution of the data. The method grows the working subset by adding other attributes into it one by one. At each step, we choose the attribute that has the highest MI with the current subset (The first intuition is used here). The MI between a variable  $X$  and a set  $\mathbf{S}$  is estimated as follows:

$$\begin{aligned} I(X; \mathbf{S}) &= \max_{Z \in \mathbf{S}} I(X; Z) \\ &= \max_{Z \in \mathbf{S}} \sum_{X, Z} P(X, Z) \log \frac{P(X, Z)}{P(X)P(Z)} \end{aligned}$$

We determine when to stop expanding the working subset by using a Bayesian statistical test called *unidimensionality test* or simply the *UD-test*. When UD-test fails, the expansion stops.

We first project original data set onto the attributes in the working subset and get a data set  $\mathbf{D}_p$ . The UD-test is done by comparing two models learned from  $\mathbf{D}_p$ : an unidimensional model  $m_1$  and a multidimensional model  $m_2$ . Model  $m_1$  and  $m_2$  are the best models, in terms of the BIC score, that contain 1 and 2 latent variables respectively. We say UD-test fails if the BIC score of  $m_2$  exceeds that of  $m_1$  by a threshold  $\delta$  and passes otherwise. i.e.,

$$BIC(m_2 | \mathbf{D}_p) - BIC(m_1 | \mathbf{D}_p) \geq \delta$$

The left hand side of the inequality is an approximation to the Bayes factor (Kass and Raftery, 1995) for the two models. When it exceeds the threshold, we conclude that correlations among the attributes cannot be appropriately modeled using one single latent variable. And these attributes in the working subset cannot all be in one sibling cluster in the final model according to the second intuition.

When UD-test fails, model  $m_2$ , which has two latent variables, gives us two potential sibling clusters. If one of them contains both the two initial attributes, we pick it as our first sibling cluster. Otherwise, we pick the one with more attributes and break ties arbitrarily. Attributes in the cluster are then removed from the data set and the process repeats to find other sibling clusters. We apply EAST algorithm to learn the  $m_1$  and  $m_2$  models. For computational efficiency, EAST is restricted to examine only models with 1 or 2 latent variables.

To illustrate the process, suppose that we start with an initial working subset which contains  $X_1$  and  $X_2$ . Two attributes  $X_3$  and  $X_4$  are successfully added. Then  $X_5$  is added. Suppose the best models  $m_1$  and  $m_2$  for  $\{X_1, X_2, X_3, X_4, X_5\}$  are as shown in Figure 2. And the BIC score of  $m_2$  exceeds that of  $m_1$  by threshold  $\delta$ . Then UD-test fails and we stop growing the subset. The sibling cluster  $\{X_1, X_2, X_4\}$  of model  $m_2$  is picked as the sibling cluster for the whole algorithm.

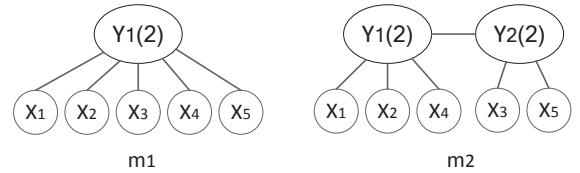


Figure 2: Model  $m_1$  and model  $m_2$  that are considered in UD-test.

### 3.2 Remaining Steps

We have described **Step 1** in Section 3.1. In the following, we describe the remaining steps.

**Step 2:** A *latent class model* (LCM) is an LTM with only one latent variable. Model  $m_1$  in Figure 2 is an example of LCM. It is a commonly used finite mixture model for discrete data. At Step 2, BI learns an LCM for each sibling cluster. It starts with an initial LCM with a binary

latent variable. The model parameters are optimized by running the EM algorithm (Koller and Friedman, 2009). Then it considers repeatedly increasing the cardinality. After each increase, model parameters are re-optimized. The process stops when the BIC score ceases to increase. **Step 3:** After the first two steps, BI obtained a collection of LCMs. In this step, we link up these LCMs in a tree formation by adding edges between the latent variables.

Chow and Liu (1968) give a well-known algorithm for learning tree-structured models among observed variables. It first estimates the MI between each pair of variables from data, then constructs a complete undirected graph with the MI values as edge weights, and finally finds the maximum spanning tree of the graph. The resulting tree model has the maximum likelihood among all tree models. Chow-Liu's algorithm can be adapted to link up the latent variables of the aforementioned LCMs. We only need to specify how the MI between two latent variables from two disjoint LCMs is to be estimated.

Given an LCM  $L_1$  with latent variable  $Y_1$ , we can first estimate probability  $P(Y_1 | L_1, \mathbf{d}_i)$  for each datacase  $\mathbf{d}_i$ . We can do this for all latent variables. In this way, we completed the data by using these LCMs. Assume another LCM  $L_2$  with latent variable  $Y_2$ . We can calculate the MI between  $Y_1$  and  $Y_2$  from the completed data. More specifically, the mutual information  $I(Y_1; Y_2)$  is computed from the following joint distribution:

$$\begin{aligned} & P(Y_1, Y_2 | \mathbf{D}, L_1, L_2) \\ &= C \sum_{i=1}^N P(Y_1 | L_1, \mathbf{d}_i) P(Y_2 | L_2, \mathbf{d}_i) \end{aligned}$$

where  $C$  is the normalization constant, and  $\mathbf{d}_i$  ( $i \in \{1, 2, \dots, N\}$ ) is the  $i$ th datacase in dataset  $\mathbf{D}$ .

**Step 4:** The sibling clusters and the cardinalities of the latent variables were determined in Steps 1 and 2. Each of those decisions was made in the context of a small number of attributes. In Step 4, BI tries to detect the possible mistakes made in those steps. More specifically, BI checks each attribute to see whether it should

be relocated and each latent variable to see if its cardinality should be changed.

In this step, BI first optimizes the probability parameters of the model resulted from the previous step using EM algorithm. The optimized model is denoted by  $\hat{m}$ . Then, similar to Step 3, for one latent variable  $Y_i$  in  $\hat{m}$ , we can estimate probability  $P(Y_i | \hat{m}, \mathbf{d}_i)$  for each datacase  $\mathbf{d}_i$ . Some message passing algorithms (Koller and Friedman, 2009) can be used to facilitate this task. We can do this for all latent variables in the whole model  $\hat{m}$ . Then the data is re-completed by using the whole model. For each observed variable  $X$  and each latent variable  $Y$ , BI computes their mutual information  $I(X; Y)$  from the completed data. Specifically, MI is computed from the following joint distribution:

$$P(X, Y | \mathbf{D}, \hat{m}) = C' \sum_{i=1}^N P(X | \mathbf{d}_i) P(Y | \hat{m}, \mathbf{d}_i)$$

where  $C'$  is the normalization constant. Let  $\hat{Y}$  be the latent variable that has the highest MI with  $X$ . If  $\hat{Y}$  is not the current parent node of  $X$  in  $\hat{m}$ , then it is deemed beneficial to relocate  $X$  from its parent node to  $\hat{Y}$ .

To determine whether a change in the cardinality of a latent variable is beneficial, BI freezes all the parameters that are not affected by the change, runs EM locally (Chen *et al.*, 2012) to optimize the parameters affected by the change, and recalculates the BIC score. The change is deemed beneficial if the BIC is increased. BI starts from the current cardinality of each latent variable and considers increasing it by one. If it is beneficial to do so, further increases are considered.

All the potential adjustments are evaluated with respect to  $\hat{m}$ . The beneficial adjustments are executed in one batch after all the evaluations. Adjustment evaluations and adjustment executions are not interleaved because that would require parameter optimization after each adjustment and hence be computationally expensive.

After model refinement, we run EM algorithm on the whole model one more time to optimize the parameters.

## 4 Empirical Results

We test our algorithm on two real world datasets. The first one is a text data known as WebKB data. It consists of web pages collected in 1997 from the computer science departments of 4 universities: Cornell, Texas, Washington and Wisconsin. Four categories of web pages are used, namely student, faculty, project and course. There are 1041 pages and the number of words was reduced to 336. Stop words and words with a low occurrence frequency are removed. The word attributes take binary values which indicate the presence or absence of the words. The second dataset is a survey data obtained from ICAC which is the anti-corruption agency of Hong Kong. This survey aims to understand public opinion towards corruption and the work performance of ICAC. After preprocessing, the dataset consists of 31 questions and 1200 records. There are missing values since some respondents do not answer all questions.

In all our experiments, parameter  $\delta$  in UDT-test is set to 3 which is a suggested threshold by Kass and Raftery (1995).

### 4.1 Results on Text Data

On the WebKB data set, BI produced an LTM with 75 latent variables. Each latent variable represents a partition. Two big questions are: (1) Are those latent variables representing meaningful partitions? (2) How can users quickly identify the desired partitions? To answer the questions, convenient tools are needed to inspect the meaning of latent variables. One such GUI tool called Lantern<sup>1</sup> is developed for this task. Given an LTM, one can easily view the tree structure, draw the information curves of each latent variable and examine *class conditional probability distributions* (CCPDs) by using Lantern.

**Information curves:** To grasp the meaning of a partition, it suffices to ask on which attributes the classes differ significantly. The information curves can help us to find the most informative attributes. For example, we draw the information curves of  $Y_{28}$  as shown in Fig-

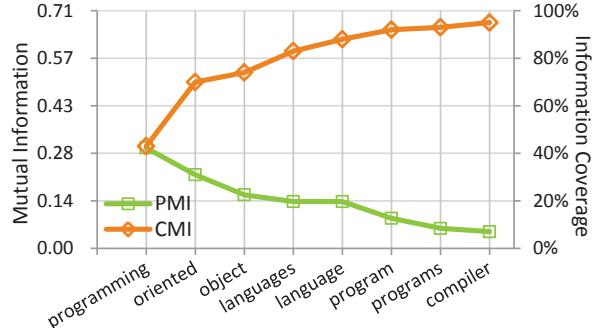


Figure 3: Information curves of latent variable  $Y_{28}$ .

ure 3. There are two curves in the figure. The lower curve shows the *pairwise mutual information* (PMI)  $I(Y_{28}; X_i)$  between  $Y_{28}$  and each attribute  $X_i$  ( $i = 1, 2, \dots, n$ ). We sorted the attributes in decreasing order of PMI. The upper curve shows the *cumulative mutual information* (CMI)  $I(Y_{28}; X_1 - X_i)$  ( $i = 2, 3, \dots, n$ ) between  $Y_{28}$  and the first  $i$  attributes. The ratio  $I(Y; X_1 - X_i)/I(Y; X_1 - X_n)$  is the *cumulative information coverage* (IC) of the first  $i$  attributes. Only the top 8 attributes are shown here. The IC of the first 8 attributes is around 95%. Intuitively, this means that the differences among the different clusters on the first 8 attributes account for 95% of the total differences. So, according to the information curves, we can say that the partition is primarily based on these attributes. Here, it is clear that the partitions represented by  $Y_{28}$  is about programming.

| cluster     | 1   | 2   | 3   | 4   |
|-------------|-----|-----|-----|-----|
| programming | 1   | .77 | .06 | .72 |
| oriented    | .16 | 0   | 0   | 1   |
| object      | .51 | .05 | .02 | .91 |
| languages   | .12 | .41 | .01 | .5  |
| language    | .81 | .34 | .05 | .61 |
| program     | .92 | .29 | .09 | .31 |
| programs    | .53 | .23 | .04 | .17 |
| compiler    | .31 | .18 | .01 | .17 |
| Size        | .04 | .21 | .69 | .06 |

**CCPDs:** To further examine how the classes differ on the attributes, we can check its *class conditional probability distributions* (CCPDs), i.e., the distributions of attributes in the class. For latent variable  $Y_{28}$ , it has 4 states, each of which represents a cluster. The table above shows part of its CCPDs. To save space, only the occurrence frequencies of the words in each cluster are shown. For example, the value in line 2 and column 3 indicates the probability that word *programming* appears in class  $Y_{28} = 2$

<sup>1</sup><http://www.cse.ust.hk/~lzheng/ltm/index.htm>

is 0.77, i.e.,  $P(\text{programming} = 1 \mid Y_{28} = 2) = 0.77$ . Probability for the absence of the word in this class is omitted. The CCPDs show that  $Y_{28}=4$  identifies web pages on *objected-oriented programming* (OOP) since the probabilities of all words are high, while  $Y_{28}=2$  identifies web pages on programming but not mentioning OOP. Those might be web pages of OOP courses and of introductory programming courses respectively.  $Y_{28}=1$  seems to correspond to web pages of other courses that involve programming, while  $Y_{28}=3$  seems to mean web pages not on programming.

Similar analysis can be done to other latent variables. Two more examples are given below. The most informative attributes are also selected based on information coverage.

| $Y_{55}$ : IC=96% |     |     |     | $Y_{57}$ : IC=100% |     |     |  |
|-------------------|-----|-----|-----|--------------------|-----|-----|--|
| cluster           | 1   | 2   | 3   | cluster            | 1   | 2   |  |
| networks          | .88 | .43 | .01 | intelligence       | .03 | .59 |  |
| communication     | .03 | .39 | .01 | artificial         | .03 | .58 |  |
| neural            | .77 | 0   | .01 | knowledge          | .03 | .6  |  |
| protocols         | 0   | .23 | 0   | ai                 | .02 | .49 |  |
| protocol          | 0   | .19 | 0   | intelligent        | .01 | .34 |  |
| intelligence      | .68 | .03 | .05 | planning           | .01 | .32 |  |
| artificial        | .66 | .03 | .05 | reasoning          | .01 | .33 |  |
| parallel          | .2  | .43 | .11 | learning           | .05 | .31 |  |
| network           | .14 | .33 | .06 | logic              | .05 | .3  |  |
| architecture      | .15 | .33 | .07 | neural             | .02 | .19 |  |
| high              | .17 | .38 | .09 | lisp               | .01 | .11 |  |
| performance       | .17 | .36 | .09 | networks           | .09 | .26 |  |
| Size              | .04 | .14 | .82 | Size               | .93 | .07 |  |

We can see that  $Y_{55}$  seems to be related with networks. The probabilities of four words, i.e. *networks*, *neural*, *artificial* and *intelligence*, are high in state 1 and low in other two states.  $Y_{55}=1$  seems to mean the networks in AI area (i.e. neural networks) and  $Y_{55}=2$  is about networks in networking and high performance computing area.  $Y_{57}=2$  seems to identify web pages belonging to faculty members who work in artificial intelligence area.

We can use Lantern to quickly identify dozens of meaningful partitions from the LTM produced by BI. To show the richness of partitions, ten other partitions are listed in Table 1. Only the words of each partition are shown. The first 4 variables  $Y_{10}$ ,  $Y_{47}$ ,  $Y_{46}$  and  $Y_{35}$  correspond to 4 universities.  $Y_6$ , same as  $Y_{28}$ , is a partition about course.  $Y_{73}$  and  $Y_{49}$  are partitions related to faculty homepages. Latent variables  $Y_{69}$ ,  $Y_{71}$  and  $Y_{59}$ , same as  $Y_{55}$  and  $Y_{57}$ , represent partitions of different research areas.

#### 4.1.1 Comparison with Alternative Methods

In this section, we compare BI with four other MPC algorithms: orthogonal projection (OP) (Cui *et al.*, 2007), singular alternative clustering (SAC) (Qi and Davidson, 2009), DK (Jain *et al.*, 2008) and EAST. For DK, OP and SAC, they were told to find two partitions each with four clusters, because it is known that there are two true class partitions each with four classes. One of true class partitions divides the web pages into four classes according to the four universities. BI has recovered the four university classes. However, they were given in the form of four latent variables instead of one. For comparability, we transformed the 4-class university partition into four logically equivalent binary class partitions. Each binary class partition divides the web pages according to whether they are from a particular university. The same transformation was applied to the other true class partition and the partitions obtained by the alternative algorithms. After the transformations, we matched up the binary class partitions with the obtained partitions and computed the *normalized mutual information* (NMI) (Strehl *et al.*, 2002) of each matched pair. The NMI between two partitions C and Y is given by  $NMI(C; Y) = I(C; Y) / \sqrt{H(C)H(Y)}$ , where  $I(\cdot)$  stands for the mutual information and  $H(\cdot)$  stands for the entropy. The results are shown in following table. The average was taken over 10 runs of the algorithms.

|            | DK      | SAC     | OP             | BI             |
|------------|---------|---------|----------------|----------------|
| course     | .43±.01 | .47±.01 | .47±.02        | <b>.63±.02</b> |
| faculty    | .18±.04 | .17±.07 | .18±.01        | <b>.30±.01</b> |
| project    | .04±.00 | .04±.00 | .05±.04        | <b>.07±.00</b> |
| student    | .18±.00 | .20±.00 | .20±.01        | <b>.25±.01</b> |
| cornell    | .22±.15 | .09±.02 | <b>.36±.24</b> | .34±.01        |
| texas      | .31±.18 | .20±.20 | .45±.23        | <b>.61±.02</b> |
| washington | .22±.13 | .41±.23 | .56±.25        | <b>.59±.12</b> |
| wisconsin  | .38±.12 | .16±.12 | .45±.13        | <b>.55±.11</b> |

We see that the NMI is the highest for BI in almost all cases. We also tested EAST on WebKB data. However, it did not finish in 14 days. In contrast, BI took only 1.1 hour.

#### 4.2 Results on Survey Data

On the survey data, BI produced an LTM with 7 latent variables. The structure of the model is shown in Figure 4. Similar as in Section 4.1,

Table 1: 10 other partitions found by BI. We cut the words when IC  $\geq 95\%$  and only nine words at most are shown.

| Universities                             |                         |                                 | Course                                       |                                                                                                 | Faculty Homepage                                                                                   |                                                                               | Research Areas                                                           |                                                                                                         |                                                                                                            |
|------------------------------------------|-------------------------|---------------------------------|----------------------------------------------|-------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|--------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| $Y_{10}$                                 | $Y_{47}$                | $Y_{46}$                        | $Y_{35}$                                     | $Y_6$                                                                                           | $Y_{73}$                                                                                           | $Y_{49}$                                                                      | $Y_{69}$                                                                 | $Y_{71}$                                                                                                | $Y_{59}$                                                                                                   |
| ithaca<br>ny<br>cornell<br>upson<br>hall | washington<br>cse<br>uw | utexas<br>texas<br>austin<br>ut | wisc<br>madison<br>wisconsin<br>dayton<br>wi | assignments<br>instructor<br>hours<br>syllabus<br>class<br>grading<br>pm<br>lecture<br>homework | journal<br>pp<br>vol<br>conference<br>proceedings<br>international<br>symposium<br>acm<br>workshop | ph<br>fax<br>research<br>professor<br>university<br>publications<br>interests | image<br>video<br>vision<br>images<br>pattern<br>digital<br>applications | management<br>database<br>systems<br>system<br>databases<br>storage<br>large<br>support<br>applications | high<br>performance<br>hardware<br>software<br>instruction<br>parallel<br>network<br>architecture<br>cache |

Table 2: The CCPDs of  $Y_1$ . The information coverage of the four attributes is around 98%. The states of Income:  $s_0$  (none),  $s_1$  (less than 4k),  $s_2$  (4-7k),  $s_3$  (7-10k),  $s_4$  (10-20k),  $s_5$  (20-40k),  $s_6$  (more than 40k). The states of Age:  $s_0$  (15-24),  $s_1$  (25-34),  $s_2$  (35-44),  $s_3$  (45-54),  $s_4$  (above 55). The states of Education:  $s_0$  (none),  $s_1$  (primary),  $s_2$  (Form 1-3),  $s_3$  (Form 4-5),  $s_4$  (Form 6-7),  $s_5$  (diploma),  $s_6$  (degree). The states of Sex:  $s_0$  (male),  $s_1$  (female).

| $P(\cdot   Y_1)$        |           | $s_0$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ |
|-------------------------|-----------|-------|-------|-------|-------|-------|-------|-------|
| $Y_1 = 1$<br>Size: 0.37 | Income    | 0     | .04   | .1    | .42   | .28   | .14   | .02   |
|                         | Age       | .05   | .35   | .39   | .17   | .03   |       |       |
|                         | Education | 0     | 0     | .04   | .41   | .09   | .09   | .37   |
|                         | Sex       | .57   | .43   |       |       |       |       |       |
| $P(\cdot   Y_1)$        |           | $s_0$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ |
| $Y_1 = 3$<br>Size: 0.22 | Income    | .11   | .17   | .25   | .31   | .07   | 0     | .1    |
|                         | Age       | 0     | .07   | .22   | .4    | .3    |       |       |
|                         | Education | .02   | .29   | .43   | .19   | .05   | .01   | 0     |
|                         | Sex       | .8    | .2    |       |       |       |       |       |

we can identify the meaning of each partition by checking the information curves and CCPDs of each latent variable. We first look at latent variables  $Y_1$  and  $Y_2$ . The most informative attributes of  $Y_1$  are *Income*, *Age*, *Education* and *Sex*. These attributes are also selected based on information curves. It is clear that  $Y_1$  partitions people based on *demographic information*. The CCPDs of  $Y_1$  are given in Table 2. It has 4 states, each of which represents a cluster. We begin with cluster  $Y_1 = 4$ . It consists people aged between 15 and 24. The average income is significantly low (78% of people in this cluster do not have income). So  $Y_1 = 4$  represents a class of *low income youngsters*. Cluster  $Y_1 = 2$  only consists of women. Between the remaining two clusters,  $Y_1 = 1$  has, on average, higher education and higher income than  $Y_1 = 3$ . Hence  $Y_1 = 1$  represents a class of people with good education and good income, while  $Y_1 = 3$  represents a class of people with poor education and average income.

The most informative attributes of  $Y_2$  are *C-NextY* (change in the level of corruption next year) and *C-PastY* (change in the level of corruption in the past year). So  $Y_2$  is about *people's view on the change of corruption level*. The CCPDs of  $Y_2$  is given in Table 3. Take  $Y_2 = 2$

| $P(\cdot   Y_1)$        |           | $s_0$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ |
|-------------------------|-----------|-------|-------|-------|-------|-------|-------|-------|
| $Y_1 = 2$<br>Size: 0.24 | Income    | .29   | .24   | .04   | 0     | 0     | 0     | .43   |
|                         | Age       | .03   | .08   | .41   | .35   | .13   |       |       |
|                         | Education | .05   | .29   | .35   | .26   | .04   | 0     | .01   |
|                         | Sex       | 0     | 1     |       |       |       |       |       |
| $P(\cdot   Y_1)$        |           | $s_0$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ |
| $Y_1 = 4$<br>Size: 0.17 | Income    | .78   | .08   | .09   | .03   | 0     | 0     | .02   |
|                         | Age       | .99   | .01   | 0     | 0     | 0     |       |       |
|                         | Education | 0     | 0     | .08   | .47   | .21   | .1    | .16   |
|                         | Sex       | .5    | .5    |       |       |       |       |       |

as an example. It represents a group of people (51%) who think the corruption level has remained the same in the past year (86%) and will remain the same next year (94%).

Table 3: The CCPDs of  $Y_2$ . The states of C-NextY:  $s_0$  (increase),  $s_1$  (decrease), and  $s_2$  (same). The states C-PastY:  $s_0$  (increased),  $s_1$  (decreased), and  $s_2$  (same).

| $P(Y_2 = 1) = 0.21$ |       |       | $P(Y_2 = 2) = 0.51$ |       |       | $P(Y_2 = 3) = 0.28$ |       |       |       |
|---------------------|-------|-------|---------------------|-------|-------|---------------------|-------|-------|-------|
| $P(\cdot   Y_2)$    | $s_0$ | $s_1$ | $s_2$               | $s_0$ | $s_1$ | $s_2$               | $s_0$ | $s_1$ | $s_2$ |
| C-NextY             | .03   | .84   | .13                 | .01   | .05   | .94                 | .79   | 0     | .21   |
| C-PastY             | .07   | .5    | .43                 | .06   | .08   | .86                 | .66   | .04   | .31   |

Similar analysis can be done to other latent variables, we will find that  $Y_3$  is a partition about people's *tolerance towards corruption*,  $Y_4$  and  $Y_5$  are about *ICAC's performance and accountability*,  $Y_6$  relates to different *corruption scene*,  $Y_7$  partitions people on their *view about economy*.

As an advantage, the relationship between partitions can be also inferred from the resulting LTM. For example, the relationship between  $Y_1$  and  $Y_2$  is revealed by the conditional probability  $P(Y_2 | Y_1)$  which is associated with the edge  $Y_1 \rightarrow Y_2$ .

| $P(Y_2   Y_1)$ | $Y_2 = 1$ | $Y_2 = 2$ | $Y_2 = 3$ |
|----------------|-----------|-----------|-----------|
| $Y_1 = 1$      | .06       | .59       | .34       |
| $Y_1 = 2$      | .30       | .42       | .28       |
| $Y_1 = 3$      | .27       | .39       | .33       |
| $Y_1 = 4$      | .34       | .60       | .06       |

We can see from the last line of above table, among the youngsters ( $Y_1 = 4$ ), 34% of them

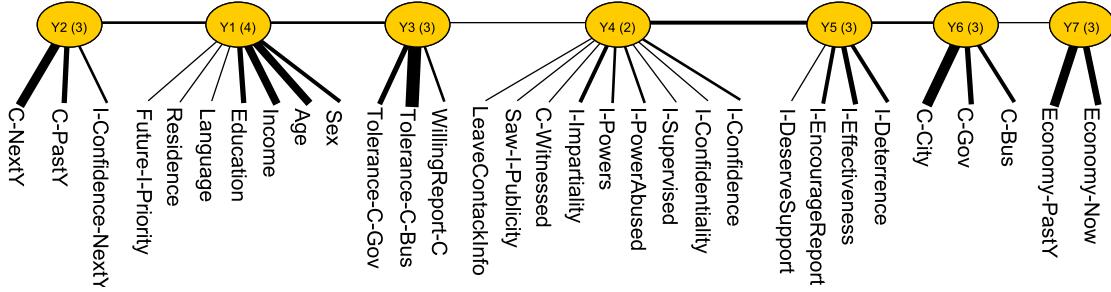


Figure 4: The structure of LTM learned by BI from the ICAC data. The width of edges represent strength of probabilistic dependence. Abbreviations: C-Corruption, I-ICAC, Y-Year, Gov-Government, Bus-Business Sector. Meanings of attributes: Tolerance-C-Gov means ‘tolerance towards corruption in the government’; C-City means ‘level of corruption in the city’; C-NextY means ‘change in the level of corruption next year’; I-Effectiveness means ‘effectiveness of ICAC’s work’; I-Powers means ‘ICAC powers’; etc.

have the positive view on the change of corruption level ( $Y_2 = 1$ ), while 6% of them have the negative view ( $Y_2 = 3$ ) on the same issue. This kind of information may be useful for users to understand the data.

For this unlabeled data, we also run EAST on it and compare the quality of models produced by both methods in terms of BIC score. EAST takes  $11312 \pm 965$  seconds to learn the model. The average is take over 10 runs. The average BIC score of the learned model is  $-26042 \pm 28$ . BI takes  $562 \pm 39$  seconds on average. The average BIC score for the learned model is  $-26096 \pm 13$ . BI archived comparative performance in much less time. For other alternative methods DK, OP and SAC, their results are not included since they can not handle datasets with missing values.

## 5 Conclusions

In this paper, we propose a greedy method for learning LTM. The new method is faster than previous algorithms. Empirical results are presented to show that our method is able to produce rich and meaningful partitions. An GUI tool is also provided to facilitate the analysis of these partitions.

## Acknowledgments

Research on this paper was supported by China National Basic Research 973 Program project No. 2011CB505101 and Guangzhou HKUST Fok Ying Tung Research Institute.

## References

Chen T, Zhang N.L, Wang Y (2008) Efficient model evaluation in the search-based approach to latent structure discovery. In: PGM-08, 57-64

- Chen T, Zhang N.L, Liu T.F, Poon K, Wang Y (2012) Model-based multidimensional clustering of categorical data. *Artif Intell* 176:2246–2269
- Choi M.J, Tan V.Y.F, Anandkumar A, Willsky A.S (2011) Learning latent tree graphical models. *Journal of Machine Learning Research* 12:1771–1812
- Chow C.K, Liu C.N (1968) Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* 14(3):462–467
- Cui Y, Fern X.Z, Dy J.G (2007) Non-redundant multi-view clustering via orthogonalization. In: ICDM-07
- Galimberti G, Soffritti G (2007) Model-based methods to identify multiple cluster structures in a data set. *CSDA-07* 52:520–536
- Gondek D, Hofmann T (2007) Non-redundant data clustering. *KAIS-07* 12(1):1–24
- Guan Y, Dy J.G, Niu D, Ghahramani Z (2010) Variational inference for nonparametric multiple clustering. In: MultiClust Workshop, KDD-2010
- Harmeling S, Williams C.K.I (2010) Greedy learning of binary latent trees. *TPAMI-10*
- Jain P, Meka R, Dhillon I.S (2008) Simultaneous unsupervised learning of disparate clusterings. *SDM-08*
- Kass R.E, Raftery A.E (1995) Bayes Factors. *Journal of the American Statistical Association* 90(430):773–795
- Koller D, Friedman N (2009) Probabilistic Graphical Models: Principles and Techniques. MIT Press
- Niu D, Dy J.G, Jordan M.I (2010) Multiple non-redundant spectral clustering views. In: ICML-10
- Poon K.M, Zhang N.L, Chen T, Yi W (2010) Variable selection in model-based clustering: To do or to facilitate. In: ICML-10
- Qi Z, Davidson I (2009) A principled and flexible framework for finding alternative clusterings. In: KDD-09
- Schwarz G (1978) Estimating the dimension of a model. *The Annals of Statistics* 6(2):461–464
- Strehl A, Ghosh J, Cardie C (2002) Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *JMLR* 3:583–617
- Zhang N.L (2004) Hierarchical latent class models for cluster analysis. *JMLR-04* 5:697–723

# Learning mixtures of polynomials from data using B-spline interpolation

Pedro L. López-Cruz, Concha Bielza and Pedro Larrañaga

Computational Intelligence Group, Departamento de Inteligencia Artificial

Facultad de Informática, Universidad Politécnica de Madrid, Spain

pedro.lcruz@upm.es, {mcbielza,pedro.larranaga}@fi.upm.es

## Abstract

Hybrid Bayesian networks efficiently encode a joint probability distribution over a set of continuous and discrete variables. Several approaches have been recently proposed for working with hybrid Bayesian networks, e.g., mixtures of truncated basis functions, mixtures of truncated exponentials or mixtures of polynomials (MoPs). We present a method for learning MoP approximations of probability densities from data using a linear combination of B-splines. Maximum likelihood estimators of the mixing coefficients of the linear combination are computed, and model selection is performed using a penalized likelihood criterion, i.e., the BIC score. Artificial examples are used to analyze the behavior of the method according to different criteria, like the quality of the approximations and the number of pieces in the MoP. Also, we study the use of the proposed method as a non-parametric density estimation technique in naive Bayes (NB) classifiers. Results on real datasets show that the non-parametric NB classifier using MoPs is comparable to the kernel density-based NB and better than Gaussian or discrete NB classifiers.

## 1 Introduction

Problems defined in hybrid domains with both continuous and discrete variables are frequently found in different fields of science. A Bayesian network (Pearl, 1988) is a kind of probabilistic graphical model which encodes a factorization of the joint probability distribution over a set of random variables. Hybrid Bayesian networks for domains with continuous and discrete variables pose a number of challenges regarding the representation of conditional probability distributions, inference, learning from data, etc.

Langseth et al. (2012) have recently proposed mixtures of truncated basis functions (MoTBFs) as a framework for representing hybrid Bayesian networks. MoTBFs generalize mixtures of truncated exponentials (MTEs) (Moral et al., 2001) and mixtures of polynomials (MoPs) (Shenoy and West, 2011). MoTBFs, MTEs and MoPs are closed under multiplication, addition and integration. Therefore, exact probabilistic inference can be performed using

the Shenoy-Shafer (1990) architecture.

Different methods have been proposed for approximating MTEs from data by using least squares (Rumí et al., 2006) or maximum likelihood (ML) estimation (Langseth et al., 2010). Recently, Langseth et al. (2012) propose methods for estimating MoTBFs by minimizing the Kullback-Leibler divergence.

In this paper, we present a method for learning MoPs directly from data using B-spline interpolation (Zong, 2006). Given a dataset, this method can be used for finding a MoP approximation of the probability density which generated the data. Our proposal ensures that the MoP is a valid density, i.e., it is continuous, non-negative and integrates to one. Previous proposals for learning MoPs assume that the mathematical expression of the generating parametric density is known (Shenoy and West, 2011) or that the true densities of the Chebyshev points are available (Shenoy, 2012). On the contrary, the proposed method only uses

the dataset without assuming any prior knowledge. First, the probability density is approximated using the B-spline interpolation method by Zong (2006), which provides ML estimators of the mixing coefficients of the linear combination of B-splines from the data. Second, the approximated B-splines are developed into a MoP function. Third, penalized likelihood scores such as the BIC score are used for performing model selection in a principled way, avoiding overfitting and models with high complexity.

The remainder of the paper is organized as follows. Section 2 reviews MoPs and the methods found in the literature for learning them. Section 3 details the proposed method for learning MoP approximations of probability densities from data. Section 4 shows the use of the proposed methods for non-parametric density estimation in naive Bayes classifiers. Sections 5 and 6 include the experimental evaluation of the proposed methods. Finally, Section 7 ends with conclusions and future work.

## 2 Mixtures of polynomials

Let  $X$  be a one-dimensional random variable with probability density  $f_X(x)$ . A MoP approximation of  $f_X(x)$  over a closed domain  $\Omega_X = [\omega_1, \omega_2] \subset \mathbb{R}$  is a  $L$ -piece  $d$ -degree piecewise function of the form (Shenoy and West, 2011)

$$\varphi_X(x) = \begin{cases} p_i(x) & \text{for } x \in A_i, i = 1, \dots, L \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where  $p_i(x)$  is a polynomial function  $b_{0i} + b_{1i}x + b_{2i}x^2 + \dots + b_{di}x^d$ ,  $\{b_{0i}, \dots, b_{di}\}$  are constants and  $A_1, \dots, A_L$  are disjoint intervals in  $\Omega_X$  which do not depend on  $x$  with  $\Omega_X = \cup_{i=1}^L A_i$ ,  $A_i \cap A_j = \emptyset, i \neq j$ .

MoPs are closed under multiplication, integration, differentiation and addition. Therefore, exact inference can be performed with the Shenoy-Shafer algorithm. Previous works used the Taylor series expansion (TSE) (Shenoy and West, 2011) or the Lagrange interpolating polynomial (LIP) (Shenoy, 2012) for estimating  $p_i(x)$ . The mathematical expression of the probability density  $f_X(x)$  needs to be

known for computing the TSE. However, real data might not fit any known parametric density, so TSE cannot be used in practice. Similarly, Shenoy (2012) proposes estimating  $p_i(x)$  as the LIP over the Chebyshev points defined in  $A_i$ . However, the true probability densities of the Chebyshev points in each  $A_i$  need to be known or estimated beforehand.

## 3 Learning MoPs using B-spline interpolation

B-splines or basis splines (Schoenberg, 1946) are polynomial curves which form a basis for the space of piecewise polynomial functions over a closed domain  $\Omega_X = [\omega_1, \omega_2]$  (Faux and Pratt, 1979). Zong (2006) proposed a method for finding B-spline approximations of probability density functions from data. He found a B-spline approximation of the density  $f_X(x)$  as a linear combination of  $M = L + r - 1$  B-splines

$$\varphi_X(x ; \boldsymbol{\alpha}) = \sum_{j=1}^M \alpha_j B_j^r(x), \quad (2)$$

where  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_M)$  are the mixing coefficients and  $B_j^r(x), j = 1, \dots, M$  are B-splines with order  $r$  (degree  $d = r - 1$ ).

Given a non-decreasing knot sequence of real numbers  $\boldsymbol{\delta} = (a_0, a_1, \dots, a_L)$ ,  $a_{i-1} < a_i$ , the  $j$ th B-spline  $B_j^r(x)$  with order  $r$  is written as

$$B_j^r(x) = (a_j - a_{j-r})H(x - a_{j-r}) \cdot \sum_{t=0}^{r-1} \frac{(a_{j-r+t} - x)^{r-1} H(a_{j-r+t} - x)}{w'_{j-r}(a_{j-r+t})}, \quad (3)$$

where  $w'_{j-r}(x)$  is the first derivative of  $w_{j-r}(x) = \prod_{u=0}^{r-1} (x - a_{j-r+u})$  and  $H(x)$  is the Heaviside function

$$H(x) = \begin{cases} 1 & x \geq 0, \\ 0 & x < 0. \end{cases}$$

A B-spline  $B_j^r(x)$  can be written as a MoP function (Equation (1)) with  $L$  pieces, where each piece  $p_i(x)$  is defined as the expansion of Equation (3) in the interval  $A_i = [a_{i-1}, a_i], i = 1, \dots, L$ . To define a MoP using B-spline interpolation, four elements need to be specified:

the order  $r$ , the number of knots/pieces  $L$ , the knot sequence  $\boldsymbol{\delta}$  and the mixing coefficients  $\boldsymbol{\alpha}$ . We used uniform B-splines so the knots in the sequence  $\boldsymbol{\delta}$  are equally spaced and yield intervals  $A_i$  with equal width:  $a_i - a_{i-1} = \frac{\omega_2 - \omega_1}{L}$ . MoPs are closed under multiplication and addition. Therefore, the linear combination of  $M$  B-splines with order  $r$  (Equation (2)) yields a MoP function with  $L$  pieces, where each piece  $p_i(x)$  is a polynomial with order  $r$  defined in the interval  $A_i$ :  $p_i(x) = \sum_{j=1}^M \alpha_j B_j^r(x), \forall x \in A_i = [a_{i-1}, a_i]$ .

B-splines have a number of interesting properties for learning MoP approximations of probability densities, e.g.,  $B_j^r(x)$  is right side continuous, differentiable, positive in  $(a_j, a_{j+r+1})$  and zero outside.

Given a dataset  $\mathcal{D} = \{x_1, \dots, x_N\}$  with  $N$  observations of variable  $X$ , Zong (2006) derived the following iterative formula for finding the ML estimators of the mixing coefficients,  $\hat{\boldsymbol{\alpha}}$ , in Equation (2):

$$\hat{\alpha}_j^{(q)} = \frac{1}{N c_j} \sum_{x \in \mathcal{D}} \frac{\hat{\alpha}_j^{(q-1)} B_j^r(x)}{\varphi_X(x; \hat{\boldsymbol{\alpha}}^{(q-1)})}, j = 1, \dots, M, \quad (4)$$

where  $q$  is the iteration number in the optimization process and

$$c_j = \int_{a_{j-r}}^{a_j} B_j^r(x) dx = \frac{a_j - a_{j-r}}{r}.$$

Zong (2006) showed that Equation (4) yields the only maximum of the log-likelihood of  $\mathcal{D}$  given the approximation (Equation (2)), subject to the constraints  $\sum_{j=1}^M \alpha_j c_j = 1$  and  $\alpha_j \geq 0, j = 1, \dots, M$ . These constraints ensure that  $\varphi_X(x; \hat{\boldsymbol{\alpha}})$  is a valid probability density, i.e., it is non-negative and integrates to one. The initial values  $\hat{\alpha}_j^{(0)}$  are set to  $1 / \sum_{j=1}^M c_j$ . Equation (4) iterates until  $\left| \frac{\ell^{(q)} - \ell^{(q-1)}}{\ell^{(q)}} \right| < \epsilon$ , where  $\ell^{(q)}$  is the log-likelihood of  $\mathcal{D}$  given  $\varphi_X(x; \hat{\boldsymbol{\alpha}}^{(q)})$  at iteration  $q$  of the optimization process. We used  $\epsilon = 10^{-6}$  in our experiments. The computational complexity of this optimization process is  $O(MNq_{\max})$ , where  $q_{\max}$  is the number of

iterations of Equation (4) performed until the algorithm converges.

Algorithm 1 summarizes the whole process for obtaining a MoP approximation of a probability density function using a dataset. Algorithm 1 needs the number of pieces  $L$  be specified a priori. Since the ML estimators of the mixing coefficients,  $\hat{\boldsymbol{\alpha}}$ , are computed in Equation (4), we can use a penalized likelihood score to perform model selection and find  $L$ . Here, we used the Bayesian information criterion (BIC) and selected the MoP with the highest BIC score:

$$BIC(\varphi_X(x), \mathcal{D}) = \ell(\mathcal{D} | \varphi_X(x)) - \frac{(M-1) \log N}{2}. \quad (5)$$

**Algorithm 1.** *Learning a MoP approximation of a probability density from data*

*Inputs:* A dataset  $\mathcal{D}$  with  $N$  observations, the number of pieces ( $L$ ) and the order of the polynomials ( $r$ ).

*Outputs:* A  $L$ -piece  $(r-1)$ -degree MoP approximation  $\varphi_X(x; \hat{\boldsymbol{\alpha}})$  of the probability density underlying the dataset  $\mathcal{D}$ .

*Steps:*

1. Compute the domain of the approximation  $\Omega_X = [\omega_1, \omega_2]$  where  $\omega_1 = \min_{\mathcal{D}}(X)$  and  $\omega_2 = \max_{\mathcal{D}}(X)$ .
2. Compute the knot sequence  $\boldsymbol{\delta} = (a_0, a_1, \dots, a_L)$  and define the intervals  $A_i = [a_{i-1}, a_i], i = 1, \dots, L$ .
3. Build the  $M = L + r - 1$  B-splines  $B_j^r(x)$  by applying Equation (3).
4. Compute the ML estimators of the mixing coefficients,  $\hat{\boldsymbol{\alpha}}$ , by applying Equation (4).
5. Compute the polynomials  $p_i(x)$  as the linear combination of the B-splines defined for each interval  $A_i$ , and build the MoP.
6. Normalize the MoP by dividing the coefficients of the polynomials  $p_i(x)$  by  $\int_{\Omega_X} \varphi_X(x) dx$ .

Algorithm 1 can be easily extended for finding MoP approximations of multivariate probability

densities from data using multivariate B-spline approximations as proposed by Zong (2006). Then, the conditional density of a variable  $X$  given its continuous parents  $\mathbf{Y}$  can be evaluated by dividing the multivariate MoP approximations of the joint densities  $\varphi_{X,\mathbf{Y}}(x, \mathbf{y})$  and  $\varphi_{\mathbf{Y}}(\mathbf{y})$ . However, obtaining MoP approximations of these joint densities is more challenging due to the higher number of parameters and the increasing number of instances needed to estimate them.

#### 4 Non-parametric naive Bayes classifiers using MoPs

In this section, we show how to use the proposed method as a non-parametric density estimation technique in naive Bayes (NB) classifiers. We consider a supervised classification problem with a discrete class variable  $C$  with values in  $\Omega_C = \{1, \dots, K\}$  and a vector of  $n$  continuous predictive variables  $\mathbf{X} = (X_1, \dots, X_n)$  with  $\Omega_{X_v} \subset \mathbb{R}, v = 1, \dots, n$ . The NB classifier (Minsky, 1961) models the probability of the class labels as a categorical distribution  $p_C(c), c \in \Omega_C$ . The predictive variables are assumed to be conditionally independent given the class. Here, we model the conditional densities of every predictive variable  $X_v$  given the class  $C = c$  with a MoP  $\varphi_{X_v|c}(x_v)$ . Algorithm 2 details the process for learning a NB classifier from data using MoPs.

**Algorithm 2.** *Learning NB classifiers with MoP approximations of the conditional density functions*

*Inputs:* A dataset  $\mathcal{D} = \{(\mathbf{x}_z, c_z)\}, z = 1, \dots, N$ , where  $\mathbf{x}_z = (x_{z1}, \dots, x_{zn})$ , the order  $r$  of the polynomials and the maximum number of pieces  $L_{max}$  for each MoP.

*Outputs:* The estimated probabilities  $p_C(c)$  and  $\varphi_{X_v|c}(x_v)$ .

1. For each class value  $c \in \Omega_C = \{1, \dots, K\}$ 
  - (a) Estimate  $p_C(c)$
  - (b) For each variable  $X_v \in \mathbf{X}$ 
    - i. Find  $\mathcal{D}_{v|c} = \{x_{zv} \in \mathcal{D} | c_z = c\}$ .
    - ii. For each  $L \in \{1, \dots, L_{max}\}$ :

- A. Find a MoP  $\varphi_{X_v|c}(x_v)$  from  $\mathcal{D}_{v|c}$  with  $L$  pieces (Algorithm 1).
- B. Compute  $BIC(\varphi_{X_v|c}(x_v), \mathcal{D}_{v|c})$  in Equation (5).
- iii. Select the MoP with the highest BIC score.

Once the probability distributions have been estimated with Algorithm 2, a new instance  $\mathbf{x}$  is classified by applying the maximum a posteriori rule:  $c^* = \arg \max_{c \in \Omega_C} p_C(c) \prod_{v=1}^n \varphi_{X_v|c}(x_v)$ .

#### 5 Experiments with MoP approximations

We analyzed the behavior of Algorithm 1 for building MoP approximations of probability densities from data using artificial examples. Figure 1 shows the MoPs obtained with 500 observations sampled from a Gaussian, an exponential and a mixture model. MoPs with order  $r = 3$  and  $L \in \{1, \dots, 10\}$  pieces were obtained using Algorithm 1. The MoPs obtained using the BIC score had fewer pieces than the MoPs with the highest log-likelihood. Equation (6) shows the MoP with the highest BIC score for the finite mixture distribution ( $L = 5$  in Figure 1(c)) as an example:

$$\varphi_X(x) = \begin{cases} 0.0567 + 0.1924x - 0.0627x^2 & 0 \leq x < 2 \\ 0.3246 - 0.0756x + 0.0043x^2 & 2 \leq x < 4 \\ 0.5716 - 0.1990x + 0.0197x^2 & 4 \leq x < 6 \\ -1.0265 + 0.3336x - 0.0247x^2 & 6 \leq x < 8 \\ 1.6972 - 0.3473x + 0.0179x^2 & 8 \leq x \leq 10 \end{cases} \quad (6)$$

It is easy to check that the MoP in Equation (6) is continuous for  $x \in \{2, 4, 6, 8\}$  and  $\int_0^{10} \varphi_X(x) dx = 1$ .

We studied the influence of the number of pieces  $L$  in the MoP approximations of the three densities in Figure 1. Figure 2 shows the log-likelihood and the BIC score of the MoPs with different values of  $L \in \{1, \dots, 10\}$ . In general, the log-likelihood of the MoPs increased with the number of pieces  $L$ . However, some values of  $L$  yielded lower log-likelihood values than MoPs

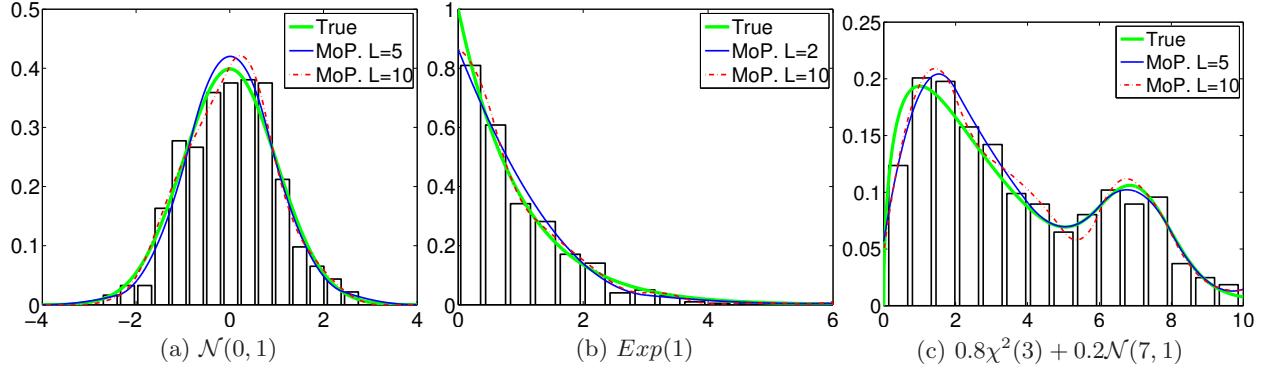


Figure 1: MoP approximations of a sample with 500 data from (a) a Gaussian, (b) an exponential and (c) a mixture of a  $\chi^2$  and a Gaussian distributions. The figure shows the true density (light solid line) and the MoP approximations with the highest BIC score (dark solid line) and the highest log-likelihood (dashed line). The histogram of the sample is also shown.

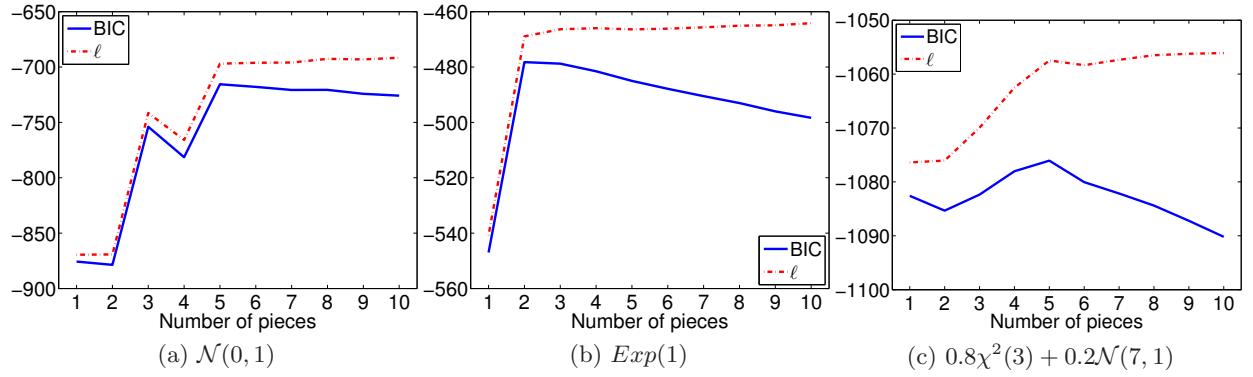


Figure 2: BIC score (solid) and log-likelihood (dashed) of the MoP approximations of the distributions in Figure 1 for different numbers of pieces  $L \in \{1, \dots, 10\}$ .

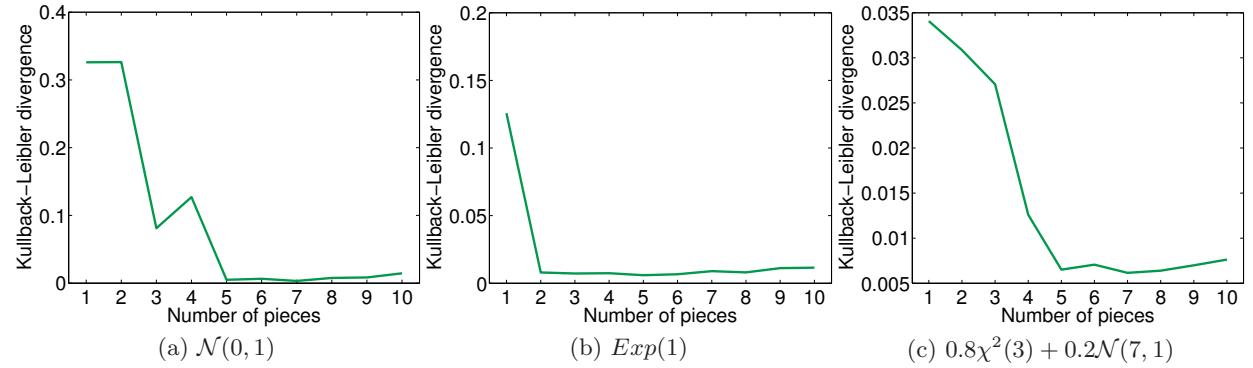


Figure 3: Kullback-Leibler divergence of the MoP approximations and the true distributions in Figure 1 for different numbers of pieces  $L \in \{1, \dots, 10\}$ .

with fewer pieces, e.g., the MoP with  $L = 4$  pieces in Figure 2(a) had a lower log-likelihood than the MoP with  $L = 3$  pieces. The domain of the MoP  $\Omega_X$  was divided into  $L$  intervals with equal width. Therefore, we conclude that a bad partition of the domain  $\Omega_X$  can yield worse MoP approximations even if a higher number of pieces is used. This highlights the importance of choosing the cut points of the intervals  $A_i$  which partition  $\Omega_X$ . We can also see that the changes in the log-likelihood of the approximations become less prominent as we increase  $L$ .

We observed that choosing the MoP with the highest log-likelihood yielded approximations with a high number of pieces which overfitted the data, e.g., Figures 1(b) and 1(c) show small oscillations in the MoPs with the highest log-likelihood ( $L = 10$ ). Figure 3 shows the Kullback-Leibler (KL) divergence of the MoPs and the true distributions for different  $L$ . We can see that the number of pieces selected using the BIC score yielded good approximations with a low KL divergence. Increasing  $L$  did not yield important reductions in the KL divergence. In fact, the KL divergence increased for high values of  $L$ , confirming that overfitting can occur in MoPs with many pieces.

## 6 Experiments with NB classifiers

We evaluated the proposed Algorithm 2 for learning a non-parametric NB classifier using MoPs. We set the order of the polynomials to  $r = 3$ , and the maximum number of pieces for a MoP to  $L_{max} = 8$ . We retrieved 14 datasets from the UCI<sup>1</sup> and KEEL<sup>2</sup> repositories and compared the proposed NB classifier (NBMoPBIC) with other NB classifiers implemented in Weka which use: Gaussian densities (NBGauss), kernel-based densities (NBKernel), Fayyad and Irani's (1993) discretization (NBFI) and equal-frequency discretization with 5 bins (NBEF5) or 10 bins (NBEF10).

Table 1 shows the mean accuracy achieved by each classifier in each dataset estimated using a stratified 10-fold cross-validation. NBMoP-

BIC achieved the best accuracy in six datasets, whereas NBKernel yielded the best accuracy in five datasets. The average ranking of the algorithms was NBKernel  $\succ$  NBMoPBIC  $\succ$  NBEF10  $\succ$  NBFI  $\succ$  NBEF5  $\succ$  NBGauss. The null hypothesis of equal performance of all algorithms was rejected at a significance level  $\alpha = 0.05$  using Friedman's test ( $p$ -value = 0.0237) and Iman-Davenport test ( $p$ -value = 0.0182).

Table 2 shows the number of datasets in which the first of the two algorithms in the tested null hypothesis ( $H_0$ ) won, tied or lost against the second algorithm. NBMoPBIC yielded better results in more datasets than the other algorithms, with the exception of NBKernel. NBKernel achieved better results in more datasets than the other algorithms. NBGauss lost in more datasets than the other algorithms.

Table 2 also includes the results of the statistical tests for finding significant differences between the algorithms. The binomial test checks whether or not the ratio of wins versus losses is significant. No significant differences between the number of wins and losses were found at a significance level  $\alpha = 0.05$ . Considering  $\alpha = 0.1$ , NBMoPBIC significantly outperformed NBGauss, whereas NBKernel significantly outperformed NBEF5. The Bergmann-Hommel post-hoc test (García and Herrera, 2008) checks all the pairwise comparisons between algorithms in all datasets. We did not find statistically significant differences between any pair of algorithms. This test ensures that the rejected null hypotheses are compatible, and this restriction makes it more difficult to find significant results when many algorithms are compared. Therefore, we also applied the Wilcoxon rank-sum non-parametric test, which compares each pair of algorithms independently taking into account all the datasets. According to this test, NBMoPBIC and NBKernel significantly outperformed NBGauss and NBEF5. Additionally, NBKernel outperformed NBFI. NBEF10 was the best performing discretization algorithm. No significant differences were found between NBMoPBIC, NBKernel and NBEF10. However, NBMoPBIC and NBKernel won in more datasets than NBEF10.

<sup>1</sup>Available at <http://archive.ics.uci.edu/ml/>

<sup>2</sup>Available at <http://www.keel.es/>

Table 1: Mean accuracy of the classifiers estimated using a stratified 10-fold cross-validation. The best result for each dataset is highlighted with boldface letters.

|              | NBMoPBIC      | NBGauss       | NBKernel      | NBFI          | NBEF5         | NBEF10        |
|--------------|---------------|---------------|---------------|---------------|---------------|---------------|
| appendicitis | <b>0.8582</b> | 0.8482        | <b>0.8582</b> | 0.8391        | 0.8200        | 0.8500        |
| fourclass    | 0.8793        | 0.7541        | <b>0.8839</b> | 0.7818        | 0.7691        | 0.8341        |
| glass2       | <b>0.9485</b> | 0.9113        | 0.9208        | 0.9251        | 0.9069        | 0.9346        |
| haberman     | 0.7295        | 0.7453        | 0.7452        | 0.7224        | 0.7388        | <b>0.7585</b> |
| ion          | <b>0.9229</b> | 0.8117        | 0.9200        | 0.8916        | 0.8859        | 0.8887        |
| iris         | <b>0.9600</b> | <b>0.9600</b> | <b>0.9600</b> | 0.9333        | 0.9333        | 0.9400        |
| liver        | 0.6453        | 0.5512        | <b>0.6832</b> | 0.5775        | 0.6394        | 0.6129        |
| newthyroid   | 0.9487        | <b>0.9632</b> | 0.9630        | 0.9489        | 0.9541        | 0.9587        |
| phoneme      | <b>0.7914</b> | 0.7600        | 0.7840        | 0.7720        | 0.7709        | 0.7707        |
| svmguide1    | 0.9432        | 0.9313        | 0.9590        | <b>0.9642</b> | 0.9601        | 0.9625        |
| vehicle      | 0.5992        | 0.4633        | 0.6134        | 0.6122        | 0.5863        | <b>0.6323</b> |
| waveform     | <b>0.8106</b> | 0.8088        | 0.8070        | 0.8078        | 0.8082        | 0.8064        |
| wdbc         | 0.9456        | 0.9331        | <b>0.9490</b> | 0.9455        | 0.9350        | 0.9473        |
| wine         | 0.9778        | 0.9778        | 0.9778        | <b>0.9833</b> | <b>0.9833</b> | 0.9667        |

Table 2: Statistical comparison of the NB classifiers. The table shows the number of datasets in which the first algorithm in the tested null hypothesis ( $H_0$ ) wins, ties or loses against the second algorithm. The  $p$ -values of the binomial, Bergmann-Hommel and Wilcoxon rank-sum tests are reported. Statistically significant results at a significance level  $\alpha = 0.05$  are highlighted in boldface.

| $H_0$               | W / T / L  | $p_{\text{Binomial}}$ | $p_{\text{Berg-Hom}}$ | $p_{\text{Wilcoxon}}$ |
|---------------------|------------|-----------------------|-----------------------|-----------------------|
| NBMoPBIC = NBKernel | 4 / 2 / 8  | 0.3877                | 1.0000                | 0.1294                |
| NBMoPBIC = NBEF10   | 8 / 0 / 6  | 0.7905                | 1.0000                | 0.3910                |
| NBMoPBIC = NBFI     | 10 / 0 / 4 | 0.1796                | 0.7423                | 0.1040                |
| NBMoPBIC = NBEF5    | 10 / 0 / 4 | 0.1796                | 0.1792                | <b>0.0353</b>         |
| NBMoPBIC = NBGauss  | 9 / 3 / 2  | 0.0654                | 0.1760                | <b>0.0244</b>         |
| NBKernel = NBEF10   | 10 / 0 / 4 | 0.1796                | 1.0000                | 0.1726                |
| NBKernel = NBFI     | 10 / 0 / 4 | 0.1796                | 0.4316                | <b>0.0203</b>         |
| NBKernel = NBEF5    | 11 / 0 / 3 | 0.0574                | 0.1000                | <b>0.0017</b>         |
| NBKernel = NBGauss  | 9 / 2 / 3  | 0.1460                | 0.0821                | <b>0.0068</b>         |
| NBEF10 = NBFI       | 9 / 0 / 5  | 0.4240                | 1.0000                | <b>0.0494</b>         |
| NBEF10 = NBEF5      | 10 / 0 / 4 | 0.1796                | 0.7423                | <b>0.0494</b>         |
| NBEF10 = NBGauss    | 10 / 0 / 4 | 0.1796                | 0.7423                | <b>0.0295</b>         |
| NBFI = NBEF5        | 8 / 2 / 4  | 0.3877                | 1.0000                | 0.3013                |
| NBFI = NBGauss      | 9 / 0 / 5  | 0.4240                | 1.0000                | 0.1531                |
| NBEF5 = NBGauss     | 8 / 0 / 6  | 0.7905                | 1.0000                | 0.2676                |

## 7 Conclusion

We have presented a method for learning MoP approximations of probability densities from data using a linear combination of B-splines. The ML estimators of the mixing coefficients of the linear combination were found and the

BIC score was used for model selection. This provided a principled way for finding the number of pieces in a MoP, which yielded accurate approximations and avoided overfitting.

The use of MoPs as a non-parametric density estimation technique for naive Bayes classifiers was also studied. NB with MoPs outperformed

Gaussian NB and discrete NB with EF5 discretization. NB with MoPs was comparable to kernel density-based NB and discrete NB with EF10 discretization. MoPs offer some advantages over kernels as non-parametric density estimators. First, MoPs provide an explicit model of the generating probability density. Second, MoPs are more efficient than kernels regarding storage and classification time because MoPs do not need to save and analyze the complete dataset to evaluate the density of a value. On the contrary, training time is higher for MoPs because parameter estimation is involved, although Equation (4) converges in few iterations (Zong, 2006).

Future work includes the extension of Algorithm 1 so that the intervals  $A_i$  do not have the same width (non-uniform B-splines). Finding the best knot sequence given a dataset is expected to reduce the number of pieces necessary to find an accurate MoP approximation of the underlying probability density. Heuristic or optimization techniques, e.g., simulated annealing or differential evolution, could be used to find estimators of the knots. Here, we only considered MoPs with order  $r = 3$ , but higher orders need to be investigated in the future. Also, extensions to more complex Bayesian classifiers which do not assume conditional independence of the predictive variables given the class will be considered, e.g., tree-augmented naive Bayes,  $k$ -dependence Bayesian classifiers, etc. Finally, the performance of the proposed method will be compared with other non-parametric techniques, e.g., MoTBFs, MTEs, etc.

## Acknowledgments

This work has been supported by the Spanish Economy and Competitiveness Ministry, Cajal Blue Brain (C080020-09), TIN2010-20900-C04-04 and Consolider Ingenio 2010-CSD2007-00018 projects. PLLC is supported by the Spanish Education Ministry (FPU AP2009-1772).

## References

I.D. Faux and M.J. Pratt. 1979. *Computational Geometry for Design and Manufacture*. Wiley.

- U. M. Fayyad and K. B. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. of the 13th IJCAI*, pages 1022–1027. Morgan Kaufmann.
- S. García and F. Herrera. 2008. An extension on “Statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694.
- H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. 2010. Parameter estimation and model selection for mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 51:485–498.
- H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. 2012. Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53:212–227.
- M. Minsky. 1961. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49:8–30.
- S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In *Proc. of the 6th ECSQARU. LNAI 2143*, pages 145–167. Springer-Verlag.
- J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.
- R. Rumí, A. Salmerón, and S. Moral. 2006. Estimating mixtures of truncated exponentials in hybrid Bayesian network. *Test*, 15(2):397–421.
- I. J. Schoenberg. 1946. Contributions to the problem of approximation of equidistant data by analytic functions. Part A: On the problem of smoothing of graduation. A first class of analytic approximation formulae. *Quarterly of Applied Mathematics*, 4:45–99.
- P. P. Shenoy and G. Shafer. 1990. Axioms for probability and belief functions propagation. In *Proc. of the 4th UAI*, pages 169–198. North-Holland.
- P. P. Shenoy and J. C. West. 2011. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52(5):641–657.
- P. P. Shenoy. 2012. Two issues in using mixtures of polynomials for inference in hybrid Bayesian networks. *International Journal of Approximate Reasoning*, 53(5):847–866.
- Z. Zong. 2006. *Information-Theoretic Methods for Estimating Complicated Probability Distributions*. Elsevier.

# A Comparison of Popular Fertility Awareness Methods to a DBN Model of the Woman’s Monthly Cycle

Anna Lupińska-Dubicka

Faculty of Computer Science, Białystok University of Technology, Poland

a.lupinska@pb.edu.pl

Marek J. Drużdżel

Decision Systems Laboratory, School of Information Sciences, University of Pittsburgh, USA,

Faculty of Computer Science, Białystok University of Technology, Poland

marek@sis.pitt.edu, m.druzdzel@pb.edu.pl

## Abstract

Fertility Awareness Methods are effective, safe, and low-cost techniques for identifying the fertile days of a menstrual cycle. In this paper, we compare the effectiveness of predicting the fertile days by a Dynamic Bayesian Network model of the monthly cycle to 11 existing Fertility Awareness Methods. We base our comparison on a real data set of 7,017 cycles collected by 881 women. We demonstrate that the DBN model is more accurate than the best modern Fertility Awareness Methods, based on the observation of mucus, marking reasonably high percentage of days of the cycle as infertile. We argue that the DBN approach offers other advantages, such as predicting the ovulation day and being able to adjust its predictions to each woman’s individual cycle.

## 1 Introduction

Fertility Awareness Methods (FAMs) are a collection of practices that help a woman know during which days of the menstrual cycle she is most likely to conceive. They are based on the observation of physiological signs of the fertile and infertile phases of the woman’s menstrual cycle. They identify the fertile window based on tracking the menstrual cycle length and/or observation of changes in one or more of the primary fertility signs, such as basal body temperature, cervical mucus, and cervix position. This knowledge can be used both to increase the chance of or to avoid pregnancy. The efficacy of FAMs to avoid pregnancy has been critically reviewed by several authors (e.g., (Arevalo et al., 2002; Frank-Herrmann et al., 2007; Howard and Stanford, 1999; Jennings and Sinai, 2001)). Additionally, FAMs can also be helpful in monitoring gynecological health and in identifying some reasons of infertility or early miscarriages.

In this paper, we examine the effectiveness

of a Dynamic Bayesian Networks (DBN) model in identifying the fertile days of a woman’s monthly cycle. Given primary fertility signs, the model predicts the time around ovulation when the probability of conception is high. Because each woman is different, the parameters of our model are based on individual characteristic of a woman’s menstrual cycle. Because every cycle can be different, we reevaluate both the structure and the parameters after each cycle. We present the results of a comparison of our DBN model to 11 popular fertility awareness methods. The data that we used in our study originate from an Italian study of daily fecundability (Colombo and Masarotto, 2000), which enrolled women from seven European centers (Milan, Verona, Lugano, Düsseldorf, Paris, London and Brussels) and from Auckland, New Zealand. From 1992 through 1996 in Europe and from 1979 through 1985 in New Zealand, 881 women collected 7,017 cycles. In our experiment, of all compared FAMs, methods based on observation of the cervical mu-

cus (i.e., the Billings Ovulation Method and the Creighton Model) perform best in the sense of indicating the shortest fertile window. Our DBN model is more accurate than the best modern Fertility Awareness Methods, based on the observation of mucus, marking reasonably high percentage of days of the cycle as infertile.

## 2 Woman's monthly cycle

The woman's monthly cycle is driven by a highly complex interaction among hormones produced by three organs of the body: the hypothalamus, the pituitary gland and the ovaries. There are four main hormones involved in the menstrual cycle process: estrogen, progesterone, follicle stimulating hormone (FSH), and lutenizing hormone (LH). The woman's monthly cycle can be divided into four phases (Figure 1): (1) menstruation, (2) the follicular phase, (3) ovulation, and (4) the luteal phase. The length of each phase may vary from woman to woman and from cycle to cycle.

In addition to measurable blood hormone levels, there are several easily accessible indicators of the phase of the cycle: raise of the basal body temperature (BBT) after ovulation, presence of the cervical mucus, and changes in position and consistency of the cervix. BBT is defined as the body temperature measured immediately after awakening and before any physical activity has been undertaken. To increase the reliability of this indicator, temperature should be measured every day at the same time. BBT follows a cyclical biphasic pattern, shifting near the ovulation from a low to a high phase. Metabolism is slower in the pre-ovulatory phase of the cycle, which results in a slightly lower body temperature. Following the ovulation, as a result of an increased level of progesterone in the body, women typically experience an increase in the BBT of at least  $0.2^{\circ}\text{C}$ . BBT remains higher until menstruation occurs or, if a woman becomes pregnant, until the end of the pregnancy. Sometimes BBT can rise due to causes other than ovulation. This atypical rise is treated as disturbance and can be caused by a change in conditions around the measurement, such as later

measurement time, lack of sleep, high stress, travel, or illness.

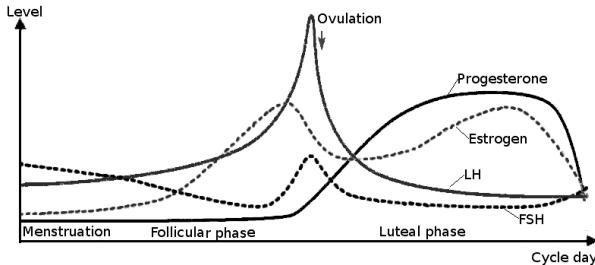


Figure 1: Levels of hormones during the phases of the woman's monthly cycle (Weschler, 2006).

As the cycle progresses, due to hormonal fluctuations, the cervical mucus increases in volume and changes its texture. Usually it is too thick for sperm to pass through. Around the time of ovulation, it becomes more elastic and less acidic, easier for sperm to penetrate. When there is no mucus or the mucus discharge is small, the day is considered infertile. There can be also a feeling of dryness around the vulva. In the luteal phase, mucus returns to its sticky stage.

## 3 Fertility Awareness Methods

Fertility Awareness Methods (FAMs) are techniques for identification of the fertile days of the monthly cycle. FAMs rely on the fact that a woman ovulates only once per menstrual cycle and she is fertile from a few days before ovulation (due to sperm life span) until after ovulation has occurred. Most menstrual cycles start with infertile days (pre-ovulatory infertility), a period of fertility and then several infertile days until the next menstruation (post-ovulatory infertility). Systems of fertility awareness identify the fertile window based on tracking menstrual cycle length and/or observation of changes in one or more of the primary fertility signs, i.e., BBT, cervical mucus, and cervix position (Weschler, 2006).

Depending on the information that they use, FAMs can be divided into three groups: (1) calendar-based methods, (2) symptoms-based methods, and (3) sympto-thermal methods.

### 3.1 Calendar-based Methods

Calendar-based methods determine both pre-ovulatory and post-ovulatory infertility taking into consideration only the length of previous menstrual cycles.

*Rhythm* method (Szymański, 2004) finds the estimated length of the pre-ovulatory infertile phase by subtracting 19 from the length of the woman's shortest cycle. Beginning of the post-ovulatory phase is determined by subtracting ten from the length of the woman's longest cycle. These calculations are updated every month using the six most recent cycles.

### 3.2 Symptoms-based Methods

Symptoms-based methods depend on the observation of changes in one or more of the primary fertility signs.

The *basal body temperature* (BBT) method rests on the fact that a woman's temperature drops 12 to 24 hours before an egg is released from her ovary. In this method, days from the first day of menstrual cycle until the third day after the BBT shift are considered fertile (Royston, 1982).

*Doering system* (DS) sets the length of the pre-ovulatory infertile phase to a woman's earliest historical day of temperature rise (in at least the previous six cycles and at most the previous 12 cycles) minus seven days. BBT shift marks the onset of post-ovulatory infertility. The BBT shift is defined as the first day in the menstrual cycle when three consecutive temperatures are above the average temperature of the last six preceding days (Barron and Fehring, 2005).

The *Billings Ovulation* (BO) method (Muzzeral, 1984) and the *Creighton Model* (CM) (Howard and Stanford, 1999) are methods recognizing and using the cervical mucus as the principal bio-marker of fertility. The first appearance of the cervical mucus is used to determine the end of the pre-ovulatory infertile phase, and its disappearance is used to determine the start of the post-ovulatory infertile phase. They differ in the method of collecting observations. The CM requires use

of toilet tissue to make observations, which are subsequently compared to a standardized mucus descriptions. The BO method instructs the woman to be aware of vulval sensations over the whole day just as she goes about her ordinary activities.

The *Two-Day* (2D) algorithm (Dunson et al., 2001; Jennings and Sinai, 2001) classifies a day as fertile if cervical secretions are present on that day or were present on the day before.

### 3.3 Sympto-thermal Methods

The sympto-thermal methods combine the calendar, the basal body temperature, and the mucus inspection methods. They can also take into consideration indicators such as breast tenderness or ovulation pains. In these methods, every primary sign of fertility is used to cross-check each other to determine the fertile and infertile days of each cycle. Any appearance of the mucus or moist sensation marks the start of the fertile period.

In the *Couple to Couple League* (CCL) method, the end of the pre-ovulatory phase is calculated by using the following formula: shortest cycle minus 21 days over the last six cycles or shortest cycle minus 20 days over the last twelve cycles, provided that there is no mucus. The first appearance of mucus marks a positive start of the fertile time. The post-ovulatory phase begins when, after the day with the most fertile mucus (mucus peak day), three consecutive temperatures are above the average temperature of the last six preceding days (Kippley and Kippley, 1996).

*Roetzer's method* (RM) considers the first six days of each cycle infertile, provided that none of the previous 12 cycles was shorter than 26 days. It finds the estimated length of the pre-ovulatory infertile phase by subtracting 20 from the length of the woman's shortest cycle. These calculations are updated every month using the 12 most recent cycles. Three days of elevated temperature after the mucus peak day mark the beginning of the post-ovulatory phase (Roetzer, 1968).

Flynn proposed the following formula (called the *English method*, EM) to determine the end

of the pre-ovulatory phase for women having knowledge about the length of their last 12 cycles: the shortest of the last 12 cycles minus 20, provided that there is no mucus. The appearance of the cervical mucus secretion starts the fertile phase. A woman who does not have records of her 12 most recent cycles, proceeds as follows: (1) in the first three cycles, the woman does not designate the pre-ovulatory infertility phase, (2) from the 4<sup>th</sup> to the 12<sup>th</sup> cycle, if none of the observed cycles is shorter than 26 days, the first 5 days of the cycle are considered infertile. For cycles shorter than 26 days, the woman should subtract 21 from the shortest cycle, (3) after the 13<sup>th</sup> cycle, the woman subtracts 20 from the shortest of the last 12 cycles. The fertile phase begins when the woman observes cervical mucus secretion (Szymański, 2004).

According to *Sensiplan* (called also the *German method*, GM), the method promoted by Arbeitsgruppe Natürliche Familienplanung in Germany (Raith et al., 1999), a woman who has just started self-observation and does not have records on the timing of her periods can consider the first five days of her menstrual cycle infertile. A woman with records of the length of her cycles determines the length of the pre-ovulatory infertility phase by subtracting 20 from the shortest cycle. A woman who has collected 12 consecutive cycles with correctly interpreted temperature curves can use the following formula: the earliest measurement of the first higher temperature minus eight gives the number of infertile days at the beginning of the cycle. The pre-ovulatory infertile phase lasts as long as the woman feels dry or does not feel anything and does not observe cervical mucus. To determine the beginning of the post-ovulatory infertility phase, women need to find the day of the BBT shift and the mucus peak day. Having marked the peak mucus and higher temperatures and adding to them three days, the end of the fertile period is determined by the symptom that appeared later.

In the method of Kramarek (called also the *Polish method*, PM) (Szymański, 2004), the end of the post-ovulatory infertility is marked by two indicators: the length of the shortest cycle

and the shortest phase of the lower temperature for the last 6-12 cycles. 6 days from the shortest phase of lower temperature should be subtracted. Between 19 and 22 days should be subtracted from the shortest cycle, depending on the length of cycles. The smaller result sets the number of days of relative infertility, provided twelve or more observed cycles. With only 6 collected cycles (up to a twelve), the result is reduced by 2 days. Appearance of feeling of moisture or any mucus begins a period of fertility. The end of the fertile period is determined by three days of elevated temperature after the mucus peak day.

## 4 Our Experiment

The term fertility awareness means that a woman knows when the fertile time of her monthly cycle starts and ends. This knowledge helps a couple in both achieving and avoiding pregnancy but can also be useful in diagnosing possible disturbances in the monthly cycle. FAMs are safe, natural, and inexpensive ways of monitoring reproductive health. They have no medical counter-indications. Their disadvantage is that they require self-discipline and can be time consuming. In our experiment, we tested the effectiveness of the existing FAMs in predicting fertile and infertile days of the monthly cycle. For this, we had a sizeable data set of real monthly cycles available and for individual records in this data set we predicted fertile and infertile days using each of the methods. While the individual records are perfect for testing, they are also used for training each of the models.

### 4.1 The Data

Our data were drawn from an Italian study of daily fecundability (Colombo and Masarotto, 2000), which enrolled women from seven European centers (Milan, Verona, Lugano, Düsseldorf, Paris, London and Brussels) and from Auckland, New Zealand. From 1992 through 1996 in Europe and from 1979 through 1985 in New Zealand, 881 women collected 7,017 cycles. To our knowledge, this is one of the most

comprehensive data sets describing woman's monthly cycle.

In each menstrual cycle, the subject was asked to record the days of her period, her basal body temperature and any disturbances such as illness, disruption of sleep or travel. She was also asked to observe and chart her cervical mucus symptoms daily during the cycle and to record every episode of coitus, and whether the couple used contraceptives or not.

The original data set include 7,017 monthly cycles collected by 881 women. However, in our analysis we included only 3,432 cycles from 236 women. We excluded all women who collected fewer than seven cycles, because a woman needs at least six cycles to become familiar with a chosen fertility awareness method. And while we wanted to compare all fertility awareness methods described in Section 3, we also excluded cycles with not uniquely identified mucus peak or the BBT shift days, because FAMs are dependent at least on one of these indicators and it is impossible to identify the fertile days of the cycle for the purpose of this analysis in cycles where the peak day is not uniquely identified. Because of current software performance limitation, we excluded women with very long cycles (longer than 40 days).

## 4.2 The Model

Our dynamic Bayesian model of woman's monthly cycle (Figure 2), combines information retrieved from BBT charting with observations of the cervical mucus secretions. It contains a variable *Phase* with four states: menstruation, follicular, ovulation, and luteal. We included three discrete observation variables: *Basal Body Temperature*, *Bleeding* and *Mucus observation*, which are readily available to any woman and also included in our data set. BBT has two possible values: lower range and higher range, representing temperature before and after the BBT shift respectively. *Bleeding* describes whether on a particular day the woman had menses or not. *Mucus observation* can be in one of four states (s1 through s4), described in detail in (Dunson et al., 2001). We modeled time explicitly as  $n$  time steps, where  $n$  is the number of

days of the longest monthly cycle of the particular woman. The model is of  $k$ -order, i.e., it contains temporal influences between 1 and  $k$ . An example of a third-order DBN is shown in Figure 2. Furthermore, while any DBN model should contain at least one first order influence, a model of order  $k$  does not need to include influences of all orders between 1 and  $k - 1$ .

Our previous paper (Łupińska-Dubicka and Drużdżel, 2011) presented the results of an experiment with a series of DBN models monitoring woman's monthly cycle. We have shown that higher order models are significantly more accurate than a static BN and first order models. However, we have also observed over-fitting and a resulting decrease of accuracy when the chosen time order or the number of temporal arcs were too high.

We found empirically that the models showed the best performance when their order was between 6 and the half of total length of the menstruation and follicular phases (this was different for each woman). Lower order would be weaker in predicting ovulation six days in advance and a higher order led to over-fitting. We have also found that it was not necessary to include influences of all orders. Skipping some of them reduced over-fitting.

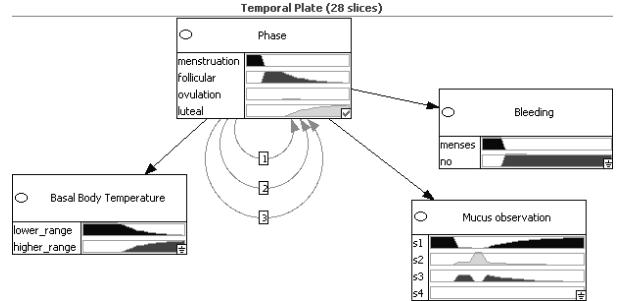


Figure 2: An example of third-order DBN model of woman's monthly cycle

In order to provide more meaningful results and to compensate for the absence of much data, we determined the initial structure of a model and its parameters based on the domain knowledge and adjusted them to a particular woman using data for her first six cycles. While a woman's body can also change over time and

with it the characteristics of the cycle, we updated the structure and parameters using not more than the last 12 woman's cycles. After each cycle, we changed the structure of a model by updating its parameters and adding or removing temporal arcs bearing in mind that first order is necessary and cannot be removed. For the last 12 cycles we calculated the minimal and most frequent day of the ovulation. Dividing these values by two we received the order of temporal arcs that should appear in model. Typically these orders were between six and nine.

### 4.3 Experiments

In our experiments, we simulated the usage of each method by women who want to become pregnant or want to avoid pregnancy, focusing on the effectiveness of each method, including our DNB model. We implemented the 11 fertility awareness method described in Section 3. To our knowledge there is no comprehensive comparison of all different FAMs.

In case of monitoring woman's monthly cycle the main goal is to predict right time of ovulation and based on it to determine the fertile window. Days inside the fertile window that were classified as infertile are *false negatives*. If the model is used to avoid pregnancy, it is critical to reduce the false negative rate to zero. Days that were marked as fertile and were outside the fertile window are *false positives*. The smaller the false positive rate, the closer the predicted day of ovulation is to the real day of ovulation, what can be helpful for couples seeking pregnancy. The number of fertile days during a menstrual cycle is difficult to specify. In our experiment, we based it on the definition given by Wilcox et al. (1995), who define the fertile window as the period between day of ovulation minus five days and day of ovulation plus one day. Many authors agree that the start of the fertile interval is strictly connected with estrogenic-type cervical mucus secretions. However, they differ in their estimates of the length of the fertile window.

We chose as our comparison criterion the percentage length of pre-ovulatory and post-ovulatory phase, and the percentage length of

the fertile window. We determined the number of fertile and infertile days in all cycles, as indicated by each of the method, and divided this number by the total length of the cycle for each woman and for each cycle. Effectively, we obtained the percentage of all days that were classified as infertile and percentage of all days that were classified as fertile. In our opinion, these two numbers (they add to 100%) are a good indication of the precision of each method. If each of these methods avoided false negatives perfectly, the larger the percentage of infertile days and the smaller the percentage of fertile days, the more precise the method and the better approximation of the ovulation day.

At every time step (i.e., every day of the cycle) our DBN model computed the most probable day of the ovulation. If a time interval between the current day and the day with the highest probability of the ovulation equaled at least six days (DBN<sub>5</sub> model, five days for life span of sperm and one more day to provide a safety margin against false negatives) or at least seven days (DBN<sub>6</sub> mode) we marked the current day as infertile. In other case the current day is the beginning of fertile period. To find the beginning of the post-ovulatory phase, our model uses the BBT shift. The third day after the BBT shift is considered as infertile.

For every implemented fertility awareness method, according to its rules, we identified the percentage length of fertile and infertile periods in all cycles. We also calculated false negative and false positive rates.

## 5 Results

Table 1 and Figure 3 show the average percentage of fertile and infertile days during a woman's monthly cycle sorted in the descending order (i.e., the longest to the shortest infertile period). The number of days in which a woman should abstain from intercourse to prevent unplanned pregnancy is larger for methods using two or more fertility indicators (i.e., symptothermal) than in symptom-based methods. As we can see, the percentage of fertile and infertile days indicated by the DBN model was close to

the mucus only based FAMs and more precise than any of the sympto-thermal methods studied. At the same time, the number of false negatives for the DBN model was close to zero for the DBN<sub>5</sub> model and zero for the DBN<sub>6</sub> model, while methods with the mucus-based methods yielding a fairly high false negative rate.

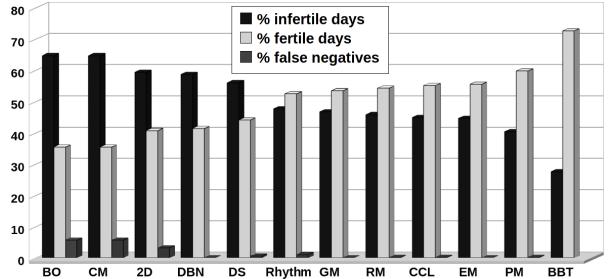


Figure 3: Graphical representation of Table 1 .

Table 1: Average percentage of fertile and infertile days and false negatives during monthly cycle for each of the compared methods (sorted by accuracy).

| Method           | % infertile days | % fertile days | % false negatives |
|------------------|------------------|----------------|-------------------|
| BO               | 64.62%           | 35.38%         | 5.68%             |
| CM               | 64.62%           | 35.38%         | 5.68%             |
| 2D               | 59.32%           | 40.68%         | 3.18%             |
| DBN <sub>5</sub> | 58.66%           | 41.34%         | 0.03%             |
| DS               | 55.95%           | 44.05%         | 0.49%             |
| DBN <sub>6</sub> | 52.41%           | 47.59%         | 0.00%             |
| Rhythm           | 47.57%           | 52.43%         | 1.03%             |
| GM               | 46.63%           | 53.37%         | 0.02%             |
| RM               | 45.76%           | 54.24%         | 0.17%             |
| CCL              | 44.90%           | 55.10%         | 0.18%             |
| EM               | 44.55%           | 55.45%         | 0.02%             |
| PM               | 40.27%           | 59.73%         | 0.17%             |
| BBT              | 27.47%           | 72.53%         | 0.00%             |

In addition to percentage of fertile and infertile days, we plotted the percentage of days of each cycle that were misclassified as infertile, i.e., percentage of false negatives. False negatives are an important measure of accuracy of a FAM, because on one hand they may lead to unplanned pregnancy and on the other hand to less likely conception in case of couples seeking pregnancy. We would like to point out that the way we determined a day to be fertile can be considered as conservative and unfair towards the modern FAMs based on mucus. Absence of mucus is an important sign of infertility and it is possible that the days that we classified as fertile were in fact infertile. However, there are women who have problems with observing and interpreting mucus secretions and methods based only on mucus are not suitable for them. And for that reason we decided to use the definition given by Wilcox et al. (1995).

## 6 Discussion

We have presented the results of a comparison of a DBN model of a woman's monthly cycle to 11 fertility awareness methods. In our analysis, methods based on observation of the cervical mucus (i.e., the BO method and CM) performed best in the sense of indicating the shortest fertile period. Our DBN model performed close to the best FAMs and showed at the same time much lower false negative rate (zero or near zero). The only other method with zero false negatives is the BBT method, but we have to take into consideration that this method determines only post-ovulatory infertility and all days before the BBT shift are considered fertile. The FAMs using two or more fertility indicators provide longer fertile window and, therefore, a lower false negatives rate, comparing to the methods based only on the cervical mucus. However, absence of cervical mucus is a strong indication of an infertile day, because without fertile type of secretions sperm cannot survive. Consequently, the false negative rate, as computed, is conservative and possibly unfair towards the mucus-based methods.

The strength of a DBN model is in that it can combine all information that a woman can collect about her cycles. It can fit an individual woman and take into consideration all fertility indicators. As our experiment showed, its performance closely matches that of the best available FAMs. Additionally, DBN model is able to predict the day of the ovulation, which can be helpful for couples seeking pregnancy.

## Acknowledgments

Our work was supported in part by the National Institute of Health under grant number U01HL101066-01. We thank Bernardo Colombo, Guido Masarotto, Fausta Ongaro, Petra Frank-Herrmann, and other investigators of the European Study of Daily Fecundability for sharing their data with us. The empirical part of the paper was performed using SMILE<sup>®</sup>, an inference engine, and GeNIE, a development environment for reasoning in graphical probabilistic models, both developed at the Decision Systems Laboratory, University of Pittsburgh, and available at <http://genie.sis.pitt.edu/>.

## References

- Marcos Arevalo, Victoria Jennings, and Irit Sinai. 2002. Efficacy of a New Method of Family Planning: the Standard Days Method. *Contraception*, 65(5):333–338.
- Mary Lee Barron and Richard J. Fehring. 2005. Basal Body Temperature Assessment: Is It Useful to Couples Seeking Pregnancy? *American Journal of Maternal Child Nursing*, 30(5):290–296.
- Bernardo Colombo and Guido Masarotto. 2000. Daily Fecundability: First Results from a New Data Base. *Demographic Research*, 3(5).
- David B. Dunson, Irit Sinai, and Bernardo Colombo. 2001. The Relationship between Cervical Secretions and the Daily Probabilities of Pregnancy Effectiveness of the TwoDay Algorithm. *Human Reproduction*, 16(11):2278–2282.
- Petra Frank-Herrmann, J. Heil, Christian Gnoth, E. Toledo, Siegfried Baur, C. Pyper, E. Jenetzky, Thomas. Strowitzki, and Günter Freundl. 2007. The Effectiveness of a Fertility Awareness Based Method to Avoid Pregnancy in Relation to a Couple's Sexual Behaviour During the Fertile Time: a Prospective Longitudinal Study. *Human Reproduction*, 22(5):1310–1319.
- Margaret P. Howard and Joseph B. Stanford. 1999. Pregnancy Probabilities during Use of the Creighton Model Fertility Care System. *Arch Fam Med*, 8(5):391–402.
- Victoria Jennings and Irit. Sinai. 2001. Further Analysis of the Theoretical Effectiveness of the TwoDay Method of Family Planning. *Contraception*, 64(3):149–53.
- John Kippley and Sheila Kippley. 1996. *The Art of Natural Family Planning (4th addition ed.)*. The Couple to Couple League, Cincinnati.
- Anna Lopińska-Dubicka and Marek J. Drużdżel. 2011. Modeling dynamic systems with memory: What is the right time-order? In *The 8th Bayesian Modelling Applications Workshop*.
- Lorna Muzzerall. 1984. The Ovulation Method of Family Planning. *Can Fam Physician*, 30:1107–9.
- Elisabeth Raith, Petra Frank, Günter Freundl, and Ursula Sottong. 1999. *Natuerliche Familienplanung Heute: Mit Ausfuehrlicher Darstellung Der Zykluscomputer Fuer Aerzte, Berater und Interessierte Anwender*. Springer.
- Joseph Roetzer. 1968. Supplemented Basal Body Temperature and Regulation of Conception. *Arch Gynakol*, 206(2):195–214.
- J. Patrick Royston. 1982. Basal Body Temperature, Ovulation and the Risk of Conception, with Special Reference to the Lifetimes of Sperm and Egg. *Biometrics*, 38(2):397–406.
- Zbigniew Szymański. 2004. *Płodność i Planowanie Rodziny*. Wydawnictwo Pomorskiej Akademii Medycznej.
- Toni Weschler. 2006. *Taking Charge of Your Fertility: The Definitive Guide to Natural Birth Control, Pregnancy Achievement, and Reproductive Health*. Collins, 10 Pap/Cdr edition.
- Allen J. Wilcox, Clarice R. Weinberg, and Donna D. Baird. 1995. Timing of Sexual Intercourse in Relation to Ovulation. Effects on the Probability of Conception, Survival of the Pregnancy, and Sex of the Baby. *New England Journal of Medicine*, 333(23):1517–1521.

# On the Importance of Elimination Heuristics in Lazy Propagation

Anders L. Madsen

HUGIN EXPERT A/S, Denmark

Department of Computer Science, Aalborg University, Denmark

alm@hugin.com

Cory J. Butz

Department of Computer Science, University of Regina, Canada

butz@cs.uregina.ca

## Abstract

Belief update in a Bayesian network using *Lazy Propagation* (LP) proceeds by message passing over a junction tree (JT). In the process of computing a message, a set of variables is eliminated. As the JT provides only a partial order on the elimination of variables, it is necessary to identify elimination orders on-line. This paper considers the importance of elimination heuristics in LP when using *Variable Elimination* (VE) as the message and single marginal computation algorithm. It considers well-known cost measures for selecting the next variable to eliminate and a new cost measure. The empirical evaluation examines different heuristics as well as sequences of cost measures, and was conducted on real-world and randomly generated Bayesian networks. The results show that for most cases performance is robust relative to the cost measure used and in some cases the elimination heuristic can have a significant impact on performance, especially for JTs that are non-optimal.

## 1 Introduction

A *Bayesian network* (BN) (Pearl, 1988) is a powerful and popular model for probabilistic inference. Its graphical nature makes it well-suited for representing complex problems where the interactions between entities represented as variables can be described using *conditional probability distributions* (CPDs). Since the computational complexity of both exact and approximate probabilistic inference in BNs are NP-hard (Cooper, 1990b) and (Dagum and Luby, 1993), we have to rely on methods that in the worst case have exponential complexity (unless P=NP). To improve feasibility of probabilistic inference using such methods, it is important to consider extensions of existing methods that can lead to better performance.

There exist two main classes of algorithms for probabilistic inference in a BN: algorithms based on message passing in a secondary com-

putational structure, i.e., a JT, (Lauritzen and Spiegelhalter, 1988; Jensen et al., 1990; Shenoy and Shafer, 1990) and direct algorithms operating on the CPDs associated with the BN, including belief propagation (Pearl, 1982; Kim and Pearl, 1983), VE (Zhang and Poole, 1994), the peeling method (Cannings et al., 1978), arc-reversal (Olmsted, 1983; Shachter, 1986), Recursive Decomposition (Cooper, 1990a) and Symbolic Probabilistic Inference (SPI) (Shachter et al., 1990; Li and D'Ambrosio, 1994; Bloemeke and Valtorta, 1998). LP (Madsen and Jensen, 1999) is a hybrid algorithm combining Shenoy-Shafer propagation and direct algorithms with the aim of exploiting independence properties induced by evidence and barren variables (Shachter, 1986).

This paper considers the importance of elimination heuristics in LP when using VE as the message and single marginal computation algorithm. The importance is assessed by an empir-

ical evaluation, including both real-world and randomly generated BNs. A total of six different heuristics for selecting the next variable to eliminate are considered along with sequences of cost measures.

The experimental results confirm the assumption that the triangulating order can have a large impact on performance and that the effectiveness of using multiple cost measures depends on the structure of the network and its JT. Multiple cost measures appear to be most effective on non-optimal JTs, and less so on (near) optimal JTs.

## 2 Preliminaries and Notation

Here preliminaries and notation are introduced.

### 2.1 Bayesian Networks

Let  $\mathcal{X} = \{X_1, \dots, X_n\}$  be a set of discrete random variables such that  $\text{dom}(X)$  is the state space of  $X$  and  $\|X\| = |\text{dom}(X)|$ . A discrete BN  $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$  over  $\mathcal{X}$  consists of an acyclic directed graph (DAG)  $G = (V, E)$  with vertices  $V$  and edges  $E$  and a set of CPDs  $\mathcal{P} = \{P(X | \text{pa}(X)) : X \in \mathcal{X}\}$ , where  $\text{pa}(X)$  denotes the parents of  $X$  in  $G$  (Pearl, 1988; Cowell et al., 1999; Kjærulff and Madsen, 2008). The BN  $\mathcal{N}$  is an encoding of a joint probability distribution over  $\mathcal{X}$

$$P(\mathcal{X}) = \prod_{i=1}^n P(X_i | \text{pa}(X_i)).$$

Belief update in  $\mathcal{N}$  is defined as the task of computing the posterior marginal  $P(X | \epsilon)$ , for each non-evidence variable  $X \in \mathcal{X} \setminus \mathcal{X}_\epsilon$  given a set of variable instantiations  $\epsilon$ , where  $\mathcal{X}_\epsilon \subseteq \mathcal{X}$  is the set of variables instantiated by  $\epsilon$ .

A potential on  $\text{dom}(\phi) = \mathcal{Y}$  is a function  $\phi$  such that  $\phi(y) \geq 0$ , for each configuration  $y \in \text{dom}(\mathcal{Y})$  and at least one  $\phi(y)$  is positive (Shafer, 1996). The *domain graph*  $G(\{\phi\}) = (V, E)$  induced by a potential  $\phi$  is defined as the graph over  $V = \text{dom}(\phi)$  with edges  $E = \{(H_1, H_2), (H_2, H_1) \mid H_1, H_2 \in \text{head}(\phi)\} \cup \{(T, H) \mid H \in \text{head}(\phi), T \in \text{tail}(\phi)\}$  where  $\text{head}(\phi)$  and  $\text{tail}(\phi)$  are the conditioned and conditioning variables of  $\text{dom}(\phi)$ , respectively.

That is,  $G$  contains an undirected edge  $(H_1, H_2)$  for each pair  $H_1, H_2 \in \text{head}(\phi)$  and a directed edge  $(T, H)$  for each pair  $T \in \text{tail}(\phi)$  and  $H \in \text{head}(\phi)$  (Madsen, 2006).

The domain graph of a set of potentials  $\Phi = \{\phi_1, \dots, \phi_m\}$  is defined as  $G(\Phi) = \bigcup_{\phi \in \Phi} G(\{\phi\})$ . In general, a domain graph will have both directed and undirected edges.

Barren variables are variables that are neither evidence nor target variables and have only barren descendants, if any (Shachter, 1986). The notion of barren variables can be extended to domain graphs (Madsen, 2006).

The weight  $w(X, Y)$  of an edge  $(X, Y)$  in a graph  $G$  is defined as  $w(X, Y) = \|X\| \cdot \|Y\|$ .

### 2.2 Lazy Propagation

LP computes all single marginals in a BN based on message passing in a JT  $T = (\mathcal{C}, \mathcal{S})$  with cliques  $\mathcal{C}$  and separators  $\mathcal{S}$ .  $T$  is constructed from  $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$  by moralisation and triangulation of  $G$ . Optimal decomposition is, however, NP-hard (Wen, 1991). Hence, the use of heuristics is justified (unless P=NP).

Once  $T$  is constructed, the CPD of each  $X \in \mathcal{X}$  is associated with a clique  $C$  such that  $\text{fa}(X) \subseteq C$ , where  $\text{fa}(X) = \{X\} \cup \text{pa}(X)$ . We let  $\Phi_C$  denote the set of CPDs associated with  $C \in \mathcal{C}$ . As part of the initialisation process CPDs are reduced to reflect the evidence  $\epsilon$ .

Belief update proceeds by passing messages between cliques over separators in two rounds relative to a root clique. Two messages are passed over each  $S \in \mathcal{S}$ ; one message in each direction. The message  $\Phi_{A \rightarrow B}$  passed from clique  $A$  to clique  $B$  consists of a set of probability potentials and it is computed by eliminating variables from a combination of potentials  $\Phi_A$  associated with  $A$  and messages received from neighboring cliques except  $B$  using VE:

$$\Phi_{A \rightarrow B} = \sum_{A \setminus B} (\Phi_A \cup \bigcup_{C \in \text{adj}(A) \setminus \{B\}} \Phi_{C \rightarrow A}),$$

where  $\text{adj}(A)$  are the cliques adjacent to  $C$ . Prior to marginalisation, barren variables (and their potentials) are removed, and only potentials for variables not separated from  $S$  by  $\epsilon$  are included in the calculation of  $\Phi_{A \rightarrow B}$ .

The structure of  $T$  induces a partial order on the elimination of  $A \setminus B$ . This means that for each message  $\Phi_{A \rightarrow B}$  (or each marginal  $P(X|\epsilon)$ ), LP has to determine the elimination order  $\sigma$  over  $A \setminus B$  on-line. In order not to jeopardise performance, it is important that the algorithm for finding  $\sigma$  is fast. As triangulation, in general, is NP-hard, we need to rely on heuristics.

### 3 Variable Elimination Heuristics

In LP, as described above, VE (Zhang and Poole, 1994) (equivalent to the *fusion* operator (Shenoy, 1997) and Bucket elimination (Dechter, 1999)) is used for computing messages and single marginals. If  $\Phi$  is a set of potentials and  $\mathcal{Y}$  is a set of variables to eliminate from  $\Phi$ , then LP uses VE to eliminate one variable  $Y \in \mathcal{Y}$  at a time by computing:

$$\begin{aligned}\phi_Y &= \sum_Y \prod_{\phi \in \Phi_Y} \phi, \\ \Phi^* &= \Phi \setminus \Phi_Y \cup \{\phi_Y\},\end{aligned}$$

where  $\Phi_Y = \{\phi \in \Phi : Y \in \text{dom}(\phi)\}$ .

In the construction of  $T$ , it may be worth spending additional resources on finding a (near) optimal triangulation as this step is performed only once and any improvement achieved will impact performance of all subsequent belief update operations. On the other hand, for the online triangulation, the aim is to produce an *efficient* elimination order  $\sigma$  *fast*. In some cases  $\sigma$  has few variables.

For variable  $X$ , we consider the cost measures  $s_d(X)$  (degree of  $X$  (or clique candidate size minus one (Rose, 1973))),  $s_{dw}(X)$  (sum of the weights of the edges adjacent to  $X$ ),  $s_{fi}(X)$  (number of fill-in-edges from eliminating  $X$  (Rose, 1973)),  $s_{fiw}(X)$  (sum of the weights of the fill-in-edges induced by eliminating  $X$  (Jensen, 2012)),  $s_{cw}(X)$  (weight of the clique candidate induced by eliminating  $X$  (Kjærulff, 1990)) and  $s_{H2}(X) = s_{cw}(X)/\|X\|$  (Cano and Moral, 1994).

Cano and Moral (1994) evaluated six heuristics H1, ..., H6. They found H1 is equivalent to  $s_{cw}$ , while H3-H6 are variants of  $s_{cw}(X)$  adjusted for the size of candidate cliques includ-

ing  $X$ , i.e., maximum size or sum of sizes. H2 is as fast to compute as H1 and produces better results, whereas H3 to H6 are expensive to compute. We consider only H2 for online triangulation.

Notice that the scores  $s_{dw}(X)$ ,  $s_{fiw}(X)$ ,  $s_{cw}(X)$  and  $s_{H2}(X)$  all use the notion of *weight* of an edge  $(X, Y)$  between  $X$  and an adjacent variable  $Y$  or a set of variables.

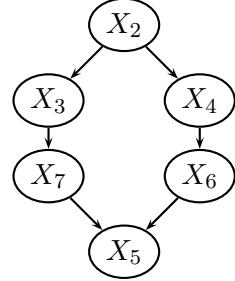


Figure 1: The domain graph for Example 1.

**Example 1.** Let  $\Phi = \{\phi(X_2), \phi(X_3 | X_2), \phi(X_4 | X_2), \phi(X_7 | X_3), \phi(X_6 | X_4), \phi(X_5 | X_7, X_8)\}$  be a set of potentials, with  $G(\Phi)$  shown in Figure 1, from which we want to compute  $\phi(X_5)$  and assume  $\|X_i\| = i$ . Different heuristics will produce different elimination orders for the computation of  $\phi(X_5)$  and assign the same score to some variables. For instance,  $s_{fi}(X_2) = s_{fi}(X_3) = s_{fi}(X_4) = 1$ ,  $s_{fiw}(X_2) = s_{fiw}(X_4) = s_{H2}(X_2) = s_{H2}(X_4) = 12$ , and  $s_{dw}(X_2) = 14$ .

Example 1 also illustrates that none of the heuristics finds the optimal order  $\sigma = (X_4, X_3, X_2, X_6, X_7)$ .

### 4 The *VarElim* Algorithm

*VarElim* is a new algorithm for eliminating a set of variables  $\mathcal{Y}$  from a set of potentials  $\Phi$  with domain graph  $G(\Phi)$ , taking as input a sequence of cost measures  $\mathcal{S} = (s_1, \dots, s_n)$  to identify the next variable to eliminate (see Algorithm 1). Starting with  $i = 1$ , if there is a tie for the best value in a cost measure  $s_i$ , the algorithm proceeds to consider  $s_{i+1}$ . The algorithm proceeds through the order to identify the variable to eliminate next.

|                                                                  |                                                                                                                                              |
|------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Data:</b>                                                     | Let $\mathcal{Y}$ be the variables to eliminate from $\Phi$ and $\mathcal{S} = (s_1, \dots, s_n)$ be an ordered sequence of score functions. |
| <b>Result:</b>                                                   | $\Phi^* = \sum_{\mathcal{Y}} \Phi$ .                                                                                                         |
| <b>begin</b>                                                     |                                                                                                                                              |
| 1 <b>foreach</b> $Y \in \mathcal{Y}$ <b>do</b>                   |                                                                                                                                              |
| <b>for</b> $i = 1$ <b>to</b> $n$ <b>do</b> Compute $s_i(Y)$      |                                                                                                                                              |
| <b>end</b>                                                       |                                                                                                                                              |
| $\mathcal{A}_0 = \mathcal{Y}$                                    |                                                                                                                                              |
| 2     <b>while</b> $ \mathcal{A}_0  > 0$ <b>do</b>               |                                                                                                                                              |
| <b>for</b> $i = 1$ <b>to</b> $n$ <b>do</b>                       |                                                                                                                                              |
| Set $\mathcal{A}_i = \arg_{Y \in \mathcal{A}_{i-1}} \min s_i(Y)$ |                                                                                                                                              |
| <b>end</b>                                                       |                                                                                                                                              |
| Eliminate $Y \in \mathcal{A}_n$                                  |                                                                                                                                              |
| Set $\mathcal{A}_0 = \mathcal{A}_0 \setminus \{Y\}$              |                                                                                                                                              |
| 5           Recompute $s_i(Y')$ for $Y' \in \text{adj}(Y)$ ,     |                                                                                                                                              |
| for $i = 1, \dots, n$                                            |                                                                                                                                              |
| <b>end</b>                                                       |                                                                                                                                              |
| <b>end</b>                                                       |                                                                                                                                              |

**Algorithm 1:** *VarElim*.

The loop in Step 1 computes the costs  $s_i(Y)$  for  $Y \in \mathcal{Y}$ . The loop in Step 2 iterates until all variables are eliminated. In Step 3,  $\mathcal{A}_i \subseteq \mathcal{A}_{i-1}$  is the set of variables with minimum cost in the score function  $s_i$ . Notice that  $|\mathcal{A}_{i-1}| = 1$  means that  $Y$  is unique. If  $|\mathcal{A}_n| = 1$ , then a unique variable has been identified. If  $|\mathcal{A}_n| > 1$ , then these variables received the same score for each  $s_i$ , for  $i = 1, 2, \dots, n$  and  $Y$  is selected at random in Step 4. In this case, the score functions in  $\mathcal{S}$  were not able to break the ties between elements of  $\mathcal{A}_n$ . In Step 4,  $Y$  is eliminated as explained in Section 3 producing an updated set of potentials  $\Phi^*$ . Notice that  $\Phi$  is updated at each iteration and  $s_i$  is recomputed in Step 5 from  $G(\Phi^*)$ .

Breaking ties in cost measures in relation to LP with AR was studied by Butz et al. (2011) as well as Madsen and Butz (2012), whereas Cano and Moral (1994) suggested tie breaking rather than selecting randomly between variables with equally good scores, and Velev and Gao (2009) suggested tie breaking by looking at the degrees of variables adjacent to the variable to eliminate.

Table 1: BNs and JT<sub>s</sub>.

| $\mathcal{N}$       | $ \mathcal{X} ,  \mathcal{C} $ | $\max_{C \in \mathcal{C}} s(C)$ | $\sum_{C \in \mathcal{C}} s(C)$ |
|---------------------|--------------------------------|---------------------------------|---------------------------------|
| Barley              | 48, 36                         | 7,257,600                       | 17,140,796                      |
| KK                  | 50, 38                         | 5,806,080                       | 14,011,466                      |
| Mildew              | 35, 29                         | 1,249,280                       | 3,400,464                       |
| OOW <sub>solo</sub> | 40, 29                         | 1,644,300                       | 4,651,788                       |
| ship-ship           | 50, 35                         | 4,032,000                       | 24,258,572                      |

Bertele and Brioschi (1972) considered the sequences  $(d, fi)$  and  $(fi, d)$  to break ties.

**Example 2.** Consider again Example 1 computing  $\phi(X_5)$  using *VarElim* with  $\mathcal{S} = (s_{fi}, s_{fiw}, s_{H2}, s_{dw})$ . The iteration in Step 3 will produce  $\mathcal{A}_1 = \{X_2, X_3, X_4, X_6, X_7\}$ ,  $\mathcal{A}_1 = \{X_2, X_3, X_4\}$ ,  $\mathcal{A}_2 = \{X_2, X_4\}$ ,  $\mathcal{A}_3 = \{X_2, X_4\}$  and  $\mathcal{A}_4 = \{X_2\}$ . This means that  $X_2$  is selected as the first variable to eliminate. The algorithm continues in the order  $\sigma = (X_3, X_4, X_6, X_7)$ .

Notice that if two consecutive heuristics  $h_i$  and  $h_{i+1}$  in  $\mathcal{S}$  always produce the same number of ties, this means that one of  $h_i$  and  $h_{i+1}$  is redundant and does not improve the tie breaking.

## 5 Empirical Evaluation

### 5.1 Experimental Setup

The experiments were performed using both real-world and randomly generated networks. We report only the results for the five real-world networks (Madsen, 2010) in Table 1, where  $s(C) = \prod_{X \in C} |\text{dom}(X)|$ . JT<sub>s</sub> have been generated using the *total weight* heuristic (Jensen, 2012) who cites (Shoikhet and Geiger, 1997). All single marginals are computed for ten different sets of evidence, for each  $|\mathcal{X}_e| = 0, \dots, n$ , where  $n = |\mathcal{X}|$ .

In the experiments, a greedy variant of *VarElim* is used. Instead of generating all candidates with the same score in each iteration, it keeps track of the best scoring variable and use the score sequence to compare and select variables.

The experiments were performed using a Java implementation (Java (TM) 2 Runtime Environment, Standard Edition (build 1.5.0\_22-b03)) running on a Linux Ubuntu (kernel 2.6.38-11-server) server with an Intel Xeon(TM) E3-1270 Processor (3.4GHz, 4C/8T and 8MB

Cache) and 32 GB RAM.

## 5.2 Different Heuristics

The aim is to analyse the impact of the elimination heuristics of Section 3 on the performance of LP. This corresponds to calling *VarElim* with  $|\mathcal{S}| = 1$ . We compare *VarElim* using min and max in Step 3 to assess the robustness of LP with respect to each cost measure.

Figure 2 shows the performance of LP using  $s_{dw}$  in Barley. The difference in performance between min and max is most significant for small  $|\mathcal{X}_e|$  and decreases as  $|\mathcal{X}_e|$  increases.

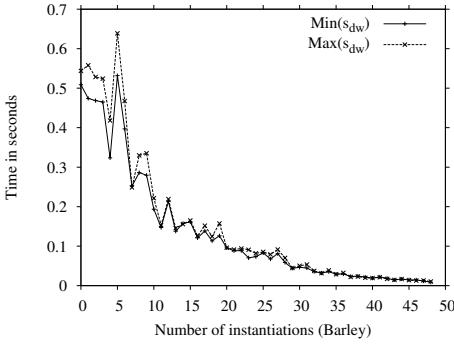


Figure 2: Belief update in Barley using  $s_{dw}$ .

Figures 3-5 show the performance of LP using  $fiw$ ,  $fi$  and  $H2$  in ship-ship, respectively. The performance difference between  $fiw$ ,  $fi$  and  $H2$  is insignificant for the min versions. This is the case across many examples considered in the tests.

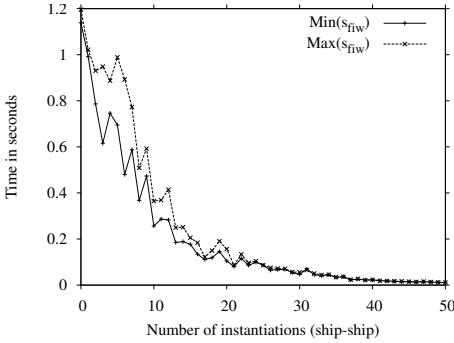


Figure 3: Belief update in ship-ship using  $fiw$ .

LP appears to be robust with respect to the heuristic used for the networks and JTs considered in the evaluation. It is important to reiterate that the JTs have been generated using

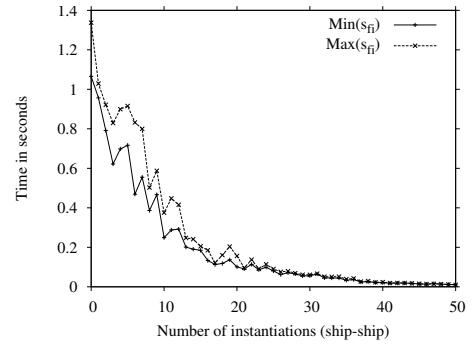


Figure 4: Belief update in ship-ship using  $fi$ .

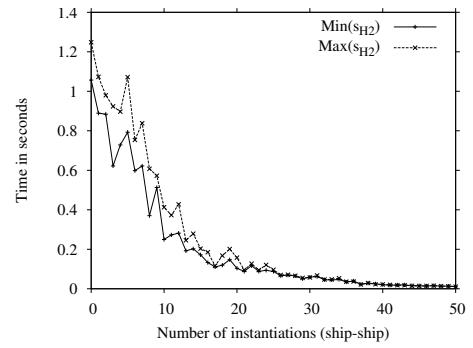


Figure 5: Belief update in ship-ship using  $H2$ .

the *total weight* heuristic that is known to often produce near optimal triangulations (Jensen, 2012). This may have a significant impact on the results.

## 5.3 Minimum or Maximum Costs

Based on the results reported in the previous section and by Kjærulff (1990), we consider four sequences of cost measures  $\mathcal{S}_0 = (s_{fiw}, s_{cw}, s_{dw}, s_{fi}, s_d)$  and its reverse sequence  $\mathcal{S}_1 = (s_d, s_{fi}, s_{dw}, s_{cw}, s_{fiw})$  as well as  $\mathcal{S}_2 = (s_{dw}, s_{fiw}, s_{cw}, s_d, s_{fi})$  and  $\mathcal{S}_3 = (s_{H2}, s_{fiw}, s_{cw}, s_{dw}, s_{fi})$ .

The aim is to analyse the impact cost measure sequences can have on performance and to assess if the potential performance improvement is robust with respect to the sequence in which the cost measures are applied. The experimental results reported in this section are focused on a comparison of *VarElim* using min in Step 3 with *VarElim* using max in Step 3.

Figure 6 shows the performance in ship-ship using  $\mathcal{S}_0$  (performance is similar for  $\mathcal{S}_1-\mathcal{S}_3$ ), and

Figure 7 shows the performance in KK using  $\mathcal{S}_1$  (performance is similar for  $\mathcal{S}_0, \mathcal{S}_2$  and  $\mathcal{S}_3$ ).

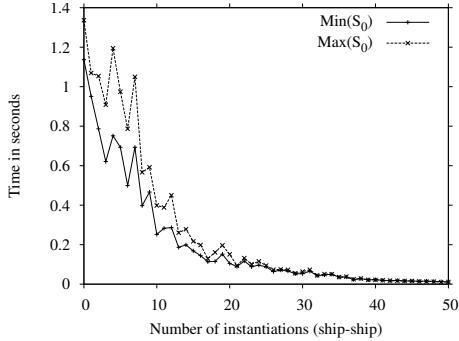


Figure 6: Belief update in ship-ship using  $\mathcal{S}_0$ .

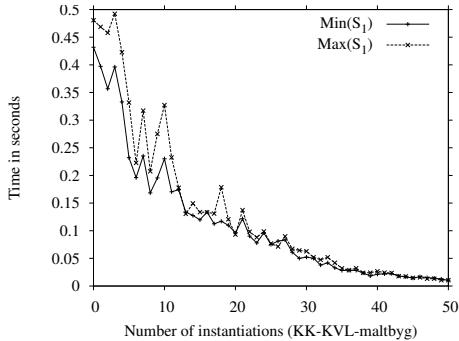


Figure 7: Belief update in KK using  $\mathcal{S}_1$ .

Figure 6 and Figure 7 illustrate that the difference in performance between using min and max in Step 3 can be significant for the orders generated. The results suggest that the partial order induced by  $T$  ensures that performance is not extremely jeopardised by using max.

#### 5.4 Best and Worst Tie Breaking

The purpose of this experiment is to analyse the potential impact cost measure sequences can have on performance, once there is a tie for minimum value in the first cost measure used, and to assess if the potential performance improvement is robust with respect to the sequence in which the cost measures are applied.

The experiment compares *VarElim* using min in Step 3 with *VarElim* using min in Step 3 on the first iteration and max in subsequent iterations. Thus, the aim is to analyse the impact of best and worst possible tie breaking once the

initial score is selected using min. Notice that best tie breaking method is equivalent to the minimum costs method.

Figure 8 shows the performance on OOW\_solo using  $\mathcal{S}_0$ . The performance is similar for  $\mathcal{S}_1 - \mathcal{S}_3$ . In fact, for most networks considered in the experiments, the difference is insignificant and can be both positive and negative.

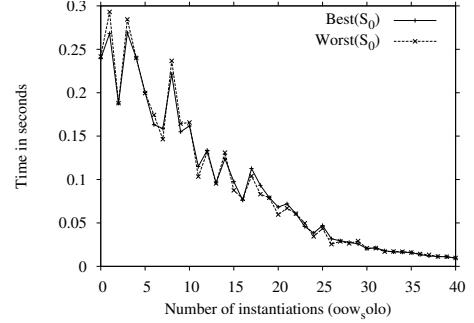


Figure 8: Belief update in OOW\_solo using  $\mathcal{S}_0$ .

Figure 9 shows the number of ties encountered for each  $s_i$  ( $i = 1, \dots, 5$ ) in  $\mathcal{S}_0$  on OOW\_solo. It also shows that the number of ties decreases as *VarElim* iterates through the scores and that for this network subsequences  $(scw, sdw)$  and  $(sf, sd)$  are not effective.

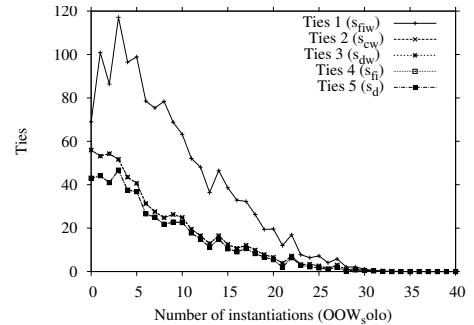


Figure 9: Score ties, OOW\_solo using  $\mathcal{S}_0$ .

In the experiments *total weight* triangulations (often optimal in clique weight) have been used for constructing  $T$ . To investigate the impact of optimal or what is believed to be near optimal triangulations, non-optimal JT<sub>s</sub> (referred to as  $T_{cs}$ ) using the minimum clique size heuristic are generated (see Table 2).

Figure 10 shows the performance in Barley using  $\mathcal{S}_0$  on  $T_{cs}$  (for  $\mathcal{S}_1 - \mathcal{S}_3$  there is almost no

Table 2: Minimum clique size JTs.

| $\mathcal{N}$ | $ \mathcal{C} $ | $\max_{C \in \mathcal{C}} s(C)$ | $\sum_{C \in \mathcal{C}} s(C)$ |
|---------------|-----------------|---------------------------------|---------------------------------|
| Barley        | 36              | 13,063,680                      | 24,970,779                      |
| Mildew        | 29              | 1,756,800                       | 4,686,212                       |
| OOW_solo      | 29              | 17,010,000                      | 32,383,477                      |

difference). Figure 11 shows the performance in Mildew using  $\mathcal{S}_1$  on  $T_{cs}$  (for  $\mathcal{S}_0$ ,  $\mathcal{S}_2$  and  $\mathcal{S}_3$  results are similar).

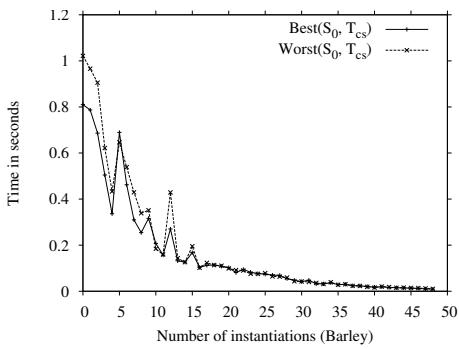
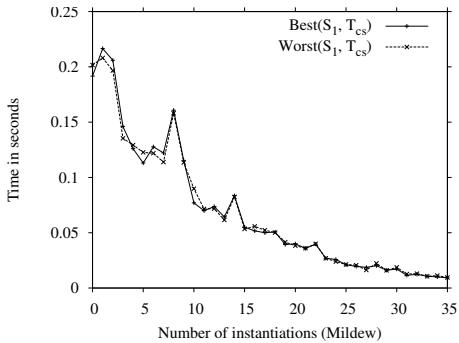
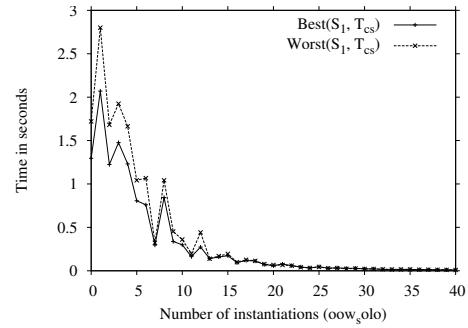
Figure 10: Barley on  $T_{cs}$  using  $\mathcal{S}_0$ .Figure 11: Mildew on  $T_{cs}$  using  $\mathcal{S}_1$ .

Figure 12 shows the performance in OOW\_solo using  $\mathcal{S}_1$  on  $T_{cs}$  (results for  $\mathcal{S}_0$  are similar). For  $\mathcal{S}_2$  and  $\mathcal{S}_3$  there is no difference and performance is at the level of  $\mathcal{S}_1$  best. As expected, LP has higher cost on  $T_{cs}$  than on  $T$ . This suggests that the initial partial order of the JT is more important for performance than the orders identified online.

## 6 Discussion and Conclusion

This paper has considered the importance of elimination heuristics in LP when using VE as

Figure 12: OOW\_solo on  $T_{cs}$  using  $\mathcal{S}_1$ .

the message and single marginal computation algorithm. The paper has introduced *VarElim* as an algorithm for eliminating a set of variables from a set of potentials using a sequence of cost measures, as opposed to using only a single cost measure. The sequence of cost measures is used for score tie breaking when identifying the variable to eliminate next.

The paper reports on a number of empirical evaluations on the performance of LP with respect to the elimination heuristics used to identify the next variable to eliminate. The results indicate that LP is robust relative to the cost measures used. This indicates that the structure of the JT is more important for performance than the online elimination heuristic. The experiments with min and max show that the time cost is increased by consistently selecting the highest scoring variable to eliminate next. The experiments with best and worst tie breaking show that breaking ties does not have a significant impact on performance when the initial JT is optimal (or believed to be near optimal), whereas the impact is more significant for less optimal JTs.

## Acknowledgments

This research was partly supported by the Spanish Ministry of Economy and Competitiveness under project TIN2010-20900-C04-01 and the European Regional Development Fund (FEDER).

## References

- U. Bertele and F. Brioschi. 1972. *Nonserial Dynamic Programming*. Academic Press.
- M. Bloemeke and M. Valtorta. 1998. A Hybrid Al-

- gorithm to Compute Marginal and Joint Beliefs in Bayesian Networks and its Complexity. In *Proc. of UAI*, pages 416–23.
- C. J. Butz, A. L. Madsen, and K. Williams. 2011. Using four cost measures to determine arc reversal orderings. In *Proc. of ECSQARU*, pages 110–121.
- C. Cannings, E. A. Thompson, and H. H. Skolnick. 1978. Probability functions on complex pedigrees. *Advances in Applied Probability*, 10:26–61.
- A. Cano and S. Moral. 1994. Heuristic algorithms for the triangulation of graphs. In *Proc. of IPMU*, pages 166–171.
- G. F. Cooper. 1990a. Bayesian Belief-Network Inference Using Recursive Decomposition. Technical Report KSL 90-05, Knowledge Systems Laboratory.
- G. F. Cooper. 1990b. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405.
- R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer.
- P. Dagum and M. Luby. 1993. Approximating probabilistic inference in Bayesian belief netwoks is NP-hard. *Artificial Intelligence*, 60:141–153.
- R. Dechter. 1999. Bucket elimination: A unifying framework for probabilistic inference. *Artificial Intelligence*, 113(1-2):41–85.
- F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. 1990. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282.
- F. Jensen. 2012. HUGIN API Reference Manual. [www.hugin.com](http://www.hugin.com). Version 7.6.
- J. H. Kim and J. Pearl. 1983. A computational model for causal and diagnostic reasoning in inference systems. In *Proc. of IJCAI*, pages 190–193.
- U. B. Kjærulff and A. L. Madsen. 2008. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer.
- U. B. Kjærulff. 1990. Graph triangulation — algorithms giving small total state space. Technical Report R 90-09, University of Aalborg, Denmark.
- S. L. Lauritzen and D. J. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *JRSS, B*, 50(2):157–224.
- Z. Li and B. D’Ambrosio. 1994. Efficient Inference in Bayes Networks As A Combinatorial Optimization Problem. *IJAR*, 11(1):55–81.
- A. L. Madsen and C. J. Butz. 2012. Ordering arc-reversal operations when eliminating variables in lazy arc propagation. Working paper, Aalborg University.
- A. L. Madsen and F. V. Jensen. 1999. Lazy propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1–2):203–245.
- A. L. Madsen. 2006. Variations Over the Message Computation Algorithm of Lazy Propagation. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, 36(3):636–648.
- A. L. Madsen. 2010. Improvements to Message Computation in Lazy Propagation. *IJAR*, 51(5):499–514.
- S. M. Olmsted. 1983. *On representing and solving decision problems*. PhD thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.
- J. Pearl. 1982. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proc. of AAAI*, pages 133–136.
- J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Series in Representation and Reasoning. Morgan Kaufmann Publishers, San Mateo, CA.
- D. J. Rose. 1973. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In *Graph Theory and Computing*, pages 183–217.
- R. Shachter, B. D’Ambrosio, and B. Del Favero. 1990. Symbolic probabilistic inference in belief networks. In *Proc. of National Conference on AI*, pages 126–131.
- R. D. Shachter. 1986. Evaluating influence diagrams. *Operations Research*, 34(6):871–882.
- G. R. Shafer. 1996. *Probabilistic Expert Systems*. SIAM.
- P. P. Shenoy and G. Shafer. 1990. Axioms for probability and belief-function propagation. In *Proc. of UAI*, pages 169–198.
- P. P. Shenoy. 1997. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *IJAR*, 17(2-3):239–263.
- K Shoikhet and D Geiger. 1997. A practical algorithm for finding optimal triangulations. In *AAAI/IAAI’97*, pages 185–190.
- M. N. Velev and P. Gao. 2009. Efficient sat techniques for absolute encoding of permutation problems: Application to hamiltonian cycles. In *Proc. of ISQED*, pages 371–376.
- W. X. Wen. 1991. Optimal decomposition of belief networks. In *Proc. of UAI*, pages 209–224.
- N. L. Zhang and D. Poole. 1994. A simple approach to Bayesian network computations. In *Proc. Canadian Conference on AI*, pages 171–178.

# A Bounded Error, Anytime, Parallel Algorithm for Exact Bayesian Network Structure Learning

Brandon Malone<sup>1,3</sup> and Changhe Yuan<sup>2,3</sup>

<sup>1</sup>Department of Computer Science, University of Helsinki, Helsinki Institute for Information Technology

<sup>2</sup>Department of Computer Science, Queens College/City University of New York

<sup>3</sup>Department of Computer Science and Engineering, Mississippi State University

brandon.malone@cs.helsinki.fi, changhe.yuan@qc.cuny.edu

## Abstract

Bayesian network structure learning is NP-hard. Several anytime structure learning algorithms have been proposed which guarantee to learn optimal networks if given enough resources. In this paper, we describe a general purpose, anytime search algorithm with bounded error that also guarantees optimality. We give an efficient, sparse representation of a key data structure for structure learning. Empirical results show our algorithm often finds better networks more quickly than state of the art methods. They also highlight accepting a small, bounded amount of suboptimality can reduce the memory and runtime requirements of structure learning by several orders of magnitude.

## 1 Introduction

Bayesian networks are a type of graphical model that are frequently used to capture relationships among variables in a domain. When a structure is unknown, we must learn it from data. In score-based structure learning, a scoring function measures the goodness of fit of a network to the data (Cooper and Herskovits, 1992; Heckerman et al., 1995). The goal is to find the structure with the optimal score. This problem is NP-hard (Chickering, 1996), so early work focused on approximation algorithms (Chickering, 2002; Moore and Wong, 2003; Teyssier and Koller, 2005). These algorithms cannot guarantee the quality of the learned structure, but they do have good anytime behavior. An anytime algorithm quickly finds a solution and improves its quality throughout the search.

Recently, dynamic programming (DP) algorithms (Koivisto and Sood, 2004; Ott et al., 2004; Singh and Moore, 2005; Silander and Myllymaki, 2006) with optimality guarantees have been developed. These algorithms learn optimal small subnetworks and grow them one leaf at a time until having the optimal network over all of the variables. Naively, though, DP algorithms store an exponential number of subnetworks. Several DP variants (Parvinainen and Koivisto, 2009; Malone et al., 2011a)

have reduced the memory requirement. None of these algorithms have anytime behavior.

Tamada *et al.* (2011) developed a parallel DP algorithm. The intuition is to group subnetworks with many overlapping computations on the same processor. They propose an indexing function which partitions variables in such a manner that provably maximizes that overlap. Consequently, it also minimizes the communication overhead. As with other DP algorithms, this algorithm does not have anytime behavior. The authors also note that, for large networks, despite minimizing communication, their MPI communication time still accounted for over 80% of the runtime.

De Campos and Ji (2011) proposed a branch and bound (BB) algorithm which searches in the space of cyclic structures to find optimal networks. They begin with a (cyclic) structure in which all variables have their optimal parents, and use a best-first search to break cycles until finding the optimal structure. To add anytime behavior, they use an approximation algorithm to initialize the search with a suboptimal network as an upper bound and sometimes vary their search strategy. As the search explores structures, it provably increases a lower bound. When the upper and lower bounds agree, the current best structure is optimal.

Several authors (Jaakkola et al., 2010; Cussens, 2011) have also developed mathematical programming (MP) algorithms to learn optimal networks. They use a series of MPs to search through a polytope with exponentially many facets to find the optimal network. These algorithms also have anytime behavior; the solution to the primal problem gives a lower bound on the score of the optimal network, and the solution to the dual can be used to decode a valid network which gives an upper bound.

Yuan *et al.* (2011) gave a shortest-path formulation of the problem. A\* and breadth-first branch and bound (Malone et al., 2011b) algorithms were developed to leverage that formulation. These algorithms also lack anytime behavior.

In this paper, we take advantage of the shortest-path formulation and dovetailing (Valenzano et al., 2010) to develop a parallel algorithm that has anytime behavior and bounds the error of learned solutions. An efficient, sparse representation for calculating search information is a key ingredient of the algorithm. Experimentally, we show that our algorithm often exhibits better anytime behavior than BB and sequential anytime search techniques. It also reveals that accepting a small, bounded amount of suboptimality in the network can reduce the memory and runtime requirements of structure learning by several orders of magnitude.

The remainder of this paper is structured as follows. Section 2 gives an overview of Bayesian network structure learning and the shortest-path formulation. Section 3 details our new algorithm. Section 4 gives experimental results in which we compare it to other state of the art algorithms.

## 2 Background

This section reviews score-based structure learning and the shortest-path formulation.

### 2.1 Learning Bayesian Network Structures

A Bayesian network is a directed acyclic graph whose vertices correspond to a set of random variables  $\mathbf{V} = \{X_1, \dots, X_n\}$ , and the arcs describe dependence relationships among the variables. The parents of  $X_i$  are called  $PA_i$ . The parameters of the network give a conditional probability distribution,  $P(X_i|PA_i)$ , for each  $X_i$ .

Given a dataset  $\mathbf{D} = \{D_1, \dots, D_N\}$ , where each  $D_i$  is a complete instantiation of  $\mathbf{V}$ , and a scoring function, the structure learning problem is to find a network structure which optimizes the scoring function for that dataset (Heckerman, 1996). We assume the use of a *decomposable* (Heckerman, 1996) scoring function, such as MDL (Lam and Bacchus, 1994) or BDe (Heckerman et al., 1995). We assume that local scores,  $score(X_i|PA_i)$ , are calculated at the beginning of the algorithm.

### 2.2 A\* Heuristic Search

A\* (Hart et al., 1968) is a state space search algorithm which guarantees to find a shortest path from a *start* node to a *goal* node. The algorithm uses an evaluation function  $f$  to measure the quality of search nodes. For a search node  $\mathbf{U}$ , the value of  $f$  is the sum of the cost from the start node to  $\mathbf{U}$ , denoted as  $g(\mathbf{U})$ , and an estimate of the cost from  $\mathbf{U}$  to a goal node, denoted as  $h(\mathbf{U})$ . That is,  $f(\mathbf{U}) = g(\mathbf{U}) + h(\mathbf{U})$ , and  $f$  gives an estimate on the cost of the shortest path from the start to the goal which passes through  $\mathbf{U}$ .

The search algorithm uses a min priority queue called *open* list to organize the search frontier. The nodes on *open* are sorted according to their  $f$  values. Initially, *open* contains only the start node. At each search step, the top node  $\mathbf{U}$  is removed from the priority queue and *expanded* by generating its successors; the expanded node is placed in a hash table called *closed*. The  $g$  cost of each successor  $\mathbf{S}$  is equal to  $g(\mathbf{U})$  plus the cost from  $\mathbf{U}$  to  $\mathbf{S}$ .

This process continues until a goal node,  $\mathbf{G}$ , is expanded. The cost of the shortest path from *start* to  $\mathbf{G}$  is  $g(\mathbf{G})$ .

An *admissible* heuristic ensures that  $h(\mathbf{U})$  is always optimistic. That is, it always underestimates the distance from  $\mathbf{U}$  to *goal*. When  $h$  is also *consistent*, the shortest path to a node is found the first time it is expanded. Therefore, if a successor is in *closed*, it is discarded. Otherwise,  $\mathbf{S}$  is added to *open*; duplicates on *open* are merged and the best cost is  $f$  is retained.

### 2.3 Shortest-path Structure Learning Formulation

Yuan *et al.* (2011) formulated the learning problem as a shortest-path finding problem. Figure 1 shows

an *implicit* state space search graph for four variables. The *start* node is the top node with the empty variable set. The *goal* node is the bottom node with the full variables set. Successors are shown by arcs in the figure. An arc from  $\mathbf{U}$  to  $\mathbf{U} \cup \{X\}$  represents adding  $X$  as a leaf to a subnetwork  $\mathbf{U}$ ; the cost of the arc is equal to the score of the optimal parent set for  $X$  out of  $\mathbf{U}$ , i.e.,

$$\text{BestScore}(X, \mathbf{U}) = \min_{PA_X \subseteq \mathbf{U}} \text{score}(X|PA_X).$$

The  $g$  cost of a node  $\mathbf{U}$  is equal to the sum of the arc costs from *start* to  $\mathbf{U}$ , and  $g(\mathbf{U})$  corresponds to the score of the optimal subnetwork over  $\mathbf{U}$ .

In this formulation, a path from *start* to *goal* induces an order on the variables, so this graph is called the *order graph*. Because each variable selects optimal parents from the existing subset when it is added, the best structure over that ordering is found by combining all of the parent sets. The shortest path corresponds to the global optimal network. In contrast to most other structure learning algorithms (Silander and Myllymaki, 2006; Jaakkola et al., 2010; de Campos and Ji, 2011; Cussens, 2011), etc., this formulation searches for the structure with the minimum score. Multiplying local scores by  $-1$  can transform between maximization and minimization.

The  $\text{BestScore}(\cdot)$  values are calculated with *parent graphs*. The parent graph for  $X$  contains all subsets of  $\mathbf{V} \setminus \{X\}$ . A parent graph containing local scores for  $X_1$  is shown in Figure 2(a). Figure 2(b) shows that, by propagating  $\text{BestScore}(\cdot)$  from subsets to supersets, the same score can be used for many nodes. Node  $\mathbf{U}$  in the parent graph of  $X$  gives the cost from  $\mathbf{U}$  to  $\mathbf{U} \cup \{X\}$  in the order graph.

A consistent heuristic was also given which estimates a lower bound on the cost from a node  $\mathbf{U}$  to *goal*. The distance from *start* to  $\mathbf{U}$  is  $g(\mathbf{U})$ , and the estimated cost to *goal* is  $h(\mathbf{U})$ . The  $f$  value, which is  $f(\mathbf{U}) = g(\mathbf{U}) + h(\mathbf{U})$ , always gives an optimistic estimate on the cost of a path through  $\mathbf{U}$ .

An A\* algorithm (Yuan et al., 2011) was shown to be an order of magnitude faster than DP; however, it requires both the parent and order graphs in RAM. Malone *et al.* (2011b) developed a breadth-first branch and bound (BFBnB) algorithm to search

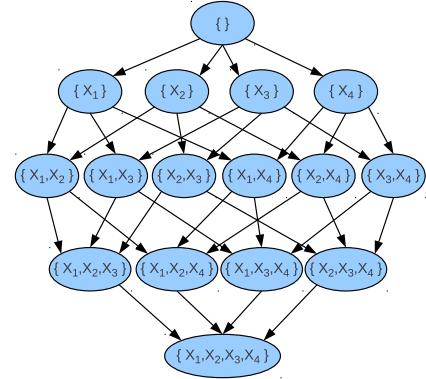


Figure 1: The order graph.

the graph layer by layer. This strategy allowed most data structures to be stored on disk and efficiently read into RAM when needed. BFBnB ran as fast as A\* and scaled to datasets with more variables. An improved heuristic function was recently proposed (Yuan and Malone, 2012) which further improved the A\* and BFBnB search algorithms.

### 3 A Bounded Error, Anytime Parallel Algorithm

The structure learning algorithms described in Section 2.3 do not have anytime behavior. They also do not take advantage of modern, multi-core architectures. In this section, we present a *bounded error, anytime parallel* search algorithm (BEAP) that has anytime behavior and bounded error for all solutions. Given enough resources, it guarantees to find the optimal network structure. It greatly improves the scalability compared to A\* and BFBnB.

We give details of the algorithm in the following subsections. Section 3.1 describes a new sparse representation for the parent graphs and integration of those into A\*. Section 3.2 presents our new parallel algorithm based on weighted A\* and dovetailing to give bounded error and anytime behavior.

#### 3.1 Sparse Parent Graph Representation

Existing formulations of the parent graphs for each variable  $X$  explicitly store  $\text{BestScore}(X, \mathbf{U})$  for all subsets  $\mathbf{V} \setminus \{X\}$ . More efficient representations (Malone et al., 2011a; Malone et al., 2011b) store  $O(C(n - 1, \frac{n}{2}))$  scores for each variable at each layer of the search. In many cases, though, the

following theorem guarantees that far fewer parent sets could be optimal (Teyssier and Koller, 2005).

**Theorem 1.** *Let  $\mathbf{U} \subset \mathbf{T}$ . If  $\text{score}(X|\mathbf{U}) < \text{score}(X|\mathbf{T})$ ,  $\mathbf{T}$  is not the optimal parent set for  $X$ .*

We adopt a different approach to leverage the pruning offered by Theorem 1. After pruning, we *sort* the remaining parent scores for each variable  $X$  in a list called  $scores_X$  in increasing (worsening) order of scores; we store the associated parent sets in an analogous list called  $parents_X$ . Because of its sorted order, the first item in  $scores_X$  gives  $\text{BestScore}(X, \mathbf{V} \setminus \{X\})$ . To find  $\text{BestScore}(X, \mathbf{U})$ , we scan the list from the beginning; the first parent set which is a subset of  $\mathbf{U}$  corresponds to  $\text{BestScore}(X, \mathbf{U})$ . Yuan and Malone (2012) describe an efficient scanning technique. In effect, this data structure allows us to efficiently operate on the pruned parent graph shown in Figure 2(c).

### 3.2 Bounded Error, Anytime, Parallel Search

In this section, we develop a *bounded error, anytime, parallel* search algorithm (BEAP). This algorithm is an example of parallel dovetailing (Valenzano et al., 2010) using Weighted A\* (Pohl, 1970; Pearl, 1984).

The bounded error aspect of our algorithm results from using Weighted A\* (WA\*) (Pohl, 1970; Pearl, 1984), which is a variant of A\* which weights the heuristic function by a factor  $\epsilon$ . That is,  $f(\mathbf{U}) = g(\mathbf{U}) + \epsilon \times h(\mathbf{U})$ . By weighting the heuristic, it is no longer admissible, so the  $f$  value for  $\mathbf{U}$  may over-estimate the cost of a path to the goal through  $\mathbf{U}$ . However, upon expanding a goal node, its cost is guaranteed to be no more than a factor of  $\epsilon$  greater than the globally optimal solution (Pearl, 1984). For example, if  $\epsilon = 1.05$ , and we expand a goal node with cost  $f$ , then the globally optimal solution is guaranteed to be no more than 5% better than  $f$ . WA\* never re-expands any nodes.

We add the anytime and parallel behavior to our algorithm by adapting parallel dovetailing (Valenzano et al., 2010). Valenzano et al. (2010) observed that many search algorithms require some sort of parameter configuration. For example, WA\* is parameterized by the weight  $\epsilon$ . Parallel dovetailing

begins by selecting a variety of parameter configurations and running each configuration in parallel processes. All configurations run until any process finds a solution, regardless of its optimality. At that point, all processes receive a message that a solution has been found, and the search stops. Because of this behavior, the version of parallel dovetailing presented is a suboptimal search strategy.

BEAP blends the advantages of WA\* and parallel dovetailing. In this algorithm we select a range of  $\epsilon$  values and run one WA\* process for each value in parallel. Unlike previous versions of dovetailing, we do not stop after finding a solution; instead, each process continues until completion. We adapt the A\* search algorithm (Yuan et al., 2011) into WA\* by passing  $\epsilon_i$  as an input to the algorithm in process  $i$ . The only algorithmic change is that, when calculating the heuristic value  $h$ , we multiply by  $\epsilon_i$ , so the  $f$  value for a node is  $f(\mathbf{U}) = g(\mathbf{U}) + \epsilon_i \times h(\mathbf{U})$ . The processes do not communicate, so they expand some of the same nodes.

The anytime behavior of the parallel algorithm results because, as the WA\* instances complete, their solutions give an upper bound on the optimal score. Typically, processes with large  $\epsilon_i$  values finish very quickly, but the scores of the learned network are high (always bounded by  $\epsilon_i$ , though). Processes with lower  $\epsilon_i$  values finish more slowly, but have better scores. Therefore, as the search progresses and WA\* instances complete, the upper bound improves. A process in which  $\epsilon = 1$ , denoted as  $\epsilon_1$ , corresponds to the unweighted, exact A\* algorithm. Therefore, the completion of that process guarantees to give the globally optimal network.

As each WA\* process completes, the quality of the solution is bounded by  $\epsilon_i$  of that process. As more processes complete, the provable bound between the optimal network and the best learned network decreases. Running  $\epsilon_1$  offers another way to bound the error. Because  $\epsilon_1$  does not weight the heuristic, no optimal network could possibly have a score better than the  $f$  value of the most recently expanded node of  $\epsilon_1$ , so that serves as a lower bound on the optimal network score. That lower bound is guaranteed to increase (or stay the same) with each node expanded in  $\epsilon_1$  because of the best-first expansion. Therefore, the ratio between the score of the best learned network and the lower bound of the

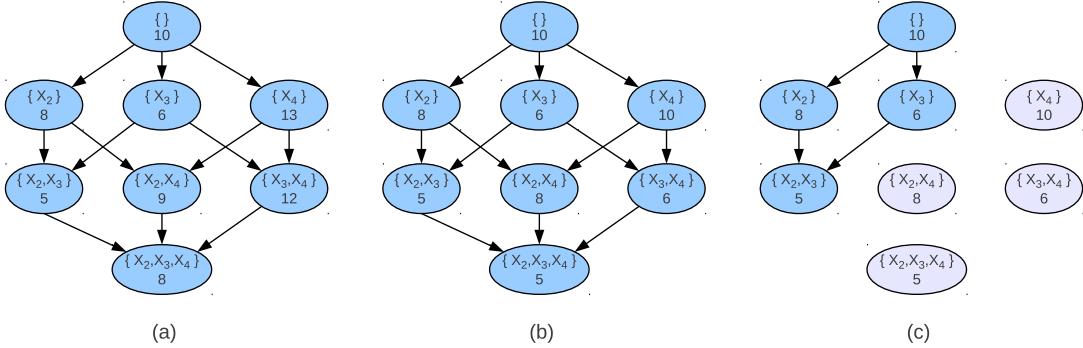


Figure 2: A sample parent graph for variable  $X_1$ . (a) The local scores,  $score(X_1, \cdot)$  for all the parent sets. (b) The optimal scores,  $BestScore(X_1, \cdot)$ , for each candidate parent set. (c) The unique optimal parent sets and their scores. Pruned parent sets are in gray. A parent set is pruned if any predecessor has a better score.

most recently expanded node of  $\epsilon_1$  also bounds the solution quality. As shown in Section 4, the ratio bound is often tighter than the bound guaranteed by  $\epsilon_i$  of the other WA\* processes.

## 4 Experimental Results

We compared BEAP to BB and another serial anytime search algorithm, Anytime WA\*.

### 4.1 Experimental Design

Anytime WA\* (AWA\*) (Hansen and Zhou, 2007) begins as the normal WA\* algorithm; however, rather than stopping the search as soon as a solution is found, AWA\* continues to expand nodes. As better paths to a goal are found, the best solution is updated, which gives the algorithm its anytime behavior. Eventually, unless interrupted, the search expands or prunes all nodes in the search space and terminates with the optimal solution. Because of the weighted heuristic, AWA\* may find a better path to a closed node. To guarantee optimality of the final solution, AWA\* must re-expand those nodes.

We evaluated BEAP on a set of benchmark datasets from the UCI repository (Frank and Asuncion, 2010). For all datasets, we removed records with missing values and discretized all variables into two states. The experiments were performed on a PC with 3.07 GHz Intel i7 processor and 16 GB of RAM. We compared BEAP to BB and a custom implementation of AWA\*. The AWA\* implementation is a straight-forward adaptation of the existing A\* algorithm (Yuan et al., 2011). Even though they are anytime algorithms, we did not compare to

any local search algorithms because they do not give an error bound. For BEAP, we used four different values of  $\epsilon$ : 1.2, 1.08, 1.04 and 1. We empirically determined that  $\epsilon > 1.2$  did not improve learning. We allowed all algorithms a total execution time of 30 minutes, not including local score calculations. BB and AWA\* are sequential, so we gave them 30 minutes of wall clock time. Since BEAP used four processes (one for each value of  $\epsilon$ ), we gave it 7.5 minutes of wall clock time, so its total time was also 30 minutes. Each BEAP process had 4 GB of RAM.

### 4.2 Node Expansion

We first evaluated the number of nodes expanded by BEAP for each value of  $\epsilon$ . The results in Figure 3 show that the algorithm typically found high quality solutions quickly. The figure also sheds insight into several characteristics of the search algorithm.

First, the searches with high  $\epsilon$  usually expand a very small number of nodes. For example, on five of the datasets, the process with  $\epsilon = 1.2$  expands the minimum number of nodes possible to find a solution ( $n + 1$ ). This takes only a fraction of a second; that processor is idle for the rest of the search. A more sophisticated scheme could be used to more fully utilize the available resources.

Second, the figure suggests that, like other combinatorial optimization problems, Bayesian network structure learning has a *critical point* (Zhang and Pemberton, 1994). A critical point for a problem is a point at which the problem difficulty undergoes a major change. Based on Figure 3, the critical point for structure learning appears to be between

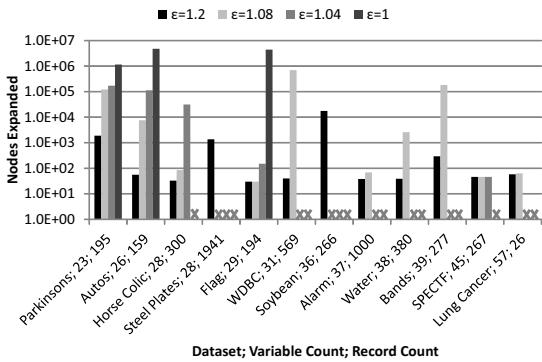


Figure 3: The number of order graph nodes expanded for each dataset and value of  $\epsilon$  by BEAP. An “X” indicates that the search did not complete within the resource restrictions.

8% and 4% of optimal. Nearly all of the instances for  $\epsilon = 1.08$  complete quickly; however, over half fail for  $\epsilon = 1.04$ . These results indicate that finding a network that is 8% of optimal is much easier than finding one that is 4% of optimal.

#### 4.3 Comparison of Anytime Behavior

We next compared the convergence and anytime behavior of BEAP to BB. As the convergence curves in Figure 4 show, BEAP finds provably high quality solutions very quickly on all of the datasets. For both *Flag* and *SPECTF*, within 2 seconds of wall clock time (8 seconds of CPU time), BEAP found networks with scores provably within 2.5% of optimal. The curves demonstrate that BEAP and BB improve error bounds differently. BB never improves its initial solution, but spends the entire 30 minutes improving its lower bound. As BEAP processes complete and  $\epsilon_1$  expands nodes, both upper and lower bounds improve.

#### 4.4 Comparison of Solution Quality

Finally, we compared the solution quality of BEAP to AWA\* and BB by comparing their upper and lower bounds. As Figure 5 shows, BEAP almost always finds a solution with a tighter error bound than the other algorithms. BEAP is the only algorithm which finds and proves the optimal structure on any of the datasets. It found tighter solutions than AWA\* because BEAP never re-expands nodes within the same process; AWA\* must re-

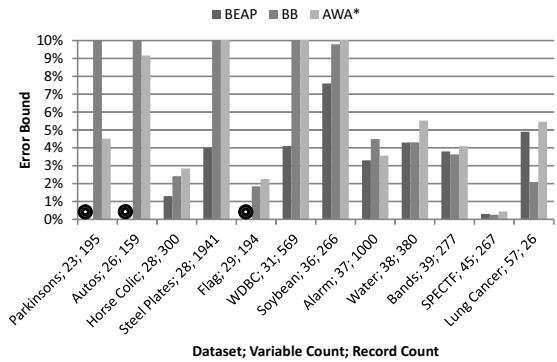


Figure 5: The error bound calculated by BEAP, AWA\* and BB. An “O” indicates that BEAP found and proved the optimal network.

expand a node each time it finds a better path to it. BB searches in the space of cyclic graphs, so these results suggest that the heuristic search formulation more effectively guides the algorithm to higher quality solutions than breaking cycles.

The bounds for BEAP are always better than the best  $\epsilon_i$  that was solved (shown in Figure 3). This shows that the bound given by the ratio between  $\epsilon_1$  and the best solution is always tighter.

For all algorithms, these results compare very favorably to those for parallel DP (Tamada et al., 2011). That algorithm took 483,874 seconds to find the optimal network for a 32 variable dataset. Of that time, 392,186 seconds were spent in MPI communication. Their algorithm also required 836.1 GB of RAM. In contrast, our algorithm used at most 16 GB, and typically less than 8 GB, which is an improvement of nearly two orders of magnitude.

## 5 Discussion

BEAP has several advantages compared to other parallel Bayesian network structure learning algorithms. First, it has very little communication overhead because each WA\* process uses a different  $\epsilon$ ; the processes do not communicate. The limited communication ensures that runtime is not wasted passing messages or waiting for synchronization, which plagued the parallel DP algorithm (Tamada et al., 2011). Second, a proper range of  $\epsilon_i$ s gives the parallel algorithm very good anytime behavior. The parallel DP algorithm (Tamada et al., 2011) does not have anytime behavior at all.

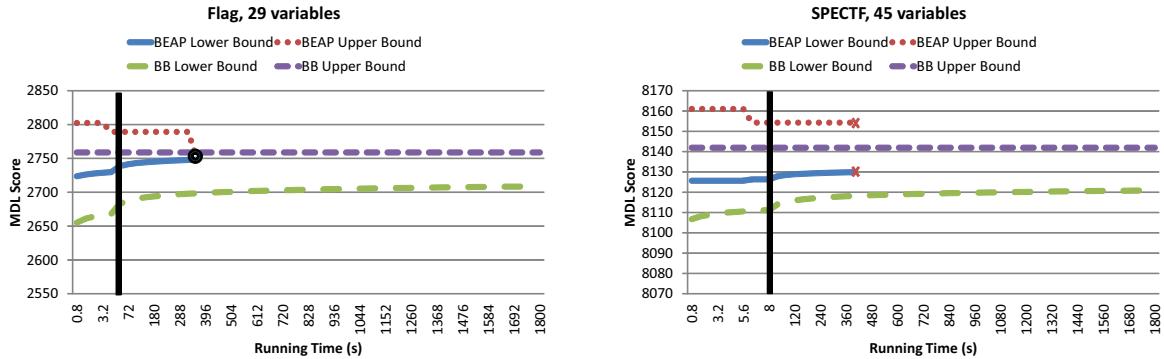


Figure 4: Convergence behavior of BEAP and BB. A black bar indicates a change in time scale. The running time is CPU, not wall clock, time. An “O” indicates that BEAP found and proved the optimal network. An “X” indicates that BEAP exceeded the RAM constraints. BB ran for the entire 30 mintues for both datasets.

BEAP also has some similarities to, and advantages over, several serial anytime search algorithms, including AWA\* (Hansen and Zhou, 2007) and Anytime Repairing A\*(ARA\*) (Likhachev et al., 2003). All three algorithms use a weighted heuristic to provably bound the error of solutions. BEAP offers advantages over these serial anytime search algorithms, though. First, BEAP re-expands nodes in parallel rather than serially. Second, in order to calculate a tighter bound than that given by  $\epsilon$ , AWA\* and ARA\* must search through the open list and calculate the true  $f$  value of each node. In contrast, BEAP simply uses the  $f$  value of the most recently expanded node of  $\epsilon_1$ . Third, unlike ARA\*, BEAP does not require any data structures other than those normally required by A\*. Like AWA\* and ARA\*, though, BEAP is a general purpose search algorithm that could be applied to any heuristic search problem, not just structure learning.

## 6 Conclusions

In this paper, we have presented a general purpose bounded error, anytime, parallel search algorithm based on weighted A\* and applied it to the Bayesian network structure learning problem. We described a sparse, efficient representation to calculate  $BestScore(\cdot)$ . Experimentally, we showed that our algorithm scales to many more variables than existing dynamic programming and shortest-path structure learning algorithms. We also showed that our algorithm often finds better solutions more quickly than existing error bounded, anytime algo-

rithms. In the future, we would like to investigate other pruning techniques to examine the critical point behavior of the structure learning problem.

## Acknowledgments

This work was supported by NSF CAREER grant IIS-0953723 and EPSCoR grant EPS-0903787.

## References

- David Maxwell Chickering. 1996. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag.
- David Maxwell Chickering. 2002. Learning equivalence classes of Bayesian-network structures. *J. Mach. Learn. Res.*, 2:445–498.
- Gregory F. Cooper and Edward Herskovits. 1992. A bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9:309–347, October.
- James Cussens. 2011. Bayesian network learning with cutting planes. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pages 153–160, Corvallis, Oregon. AUAI Press.
- Cassio P. de Campos and Qiang Ji. 2011. Efficient learning of bayesian networks using constraints. *Journal of Machine Learning Research*, 12:663–689.
- A. Frank and A. Asuncion. 2010. UCI machine learning repository.
- Eric A. Hansen and Rong Zhou. 2007. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28:267–297.
- P E Hart, N J Nilsson, and B Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions On Systems Science And Cybernetics*, 4(2):100–107.

- David Heckerman, Dan Geiger, and David M. Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. 20:197–243.
- David Heckerman. 1996. A tutorial on learning with Bayesian networks. Technical report, Learning in Graphical Models.
- Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. 2010. Learning Bayesian network structure using LP relaxations. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Mikko Koivisto and Kismat Sood. 2004. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, pages 549–573.
- Wai Lam and Fahiem Bacchus. 1994. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293.
- M. Likhachev, G. Gordon, and S. Thrun. 2003. ARA\*: Anytime A\* search with provable bounds on sub-optimality. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Proceedings of Conference on Neural Information Processing Systems (NIPS)*. MIT Press.
- Brandon Malone, Changhe Yuan, and Eric Hansen. 2011a. Memory-efficient dynamic programming for learning optimal Bayesian networks. In *Proceedings of the 25th national conference on Artifical intelligence*.
- Brandon Malone, Changhe Yuan, Eric Hansen, and Susan Bridges. 2011b. Improving the scalability of optimal Bayesian network learning with external-memory frontier breadth-first branch and bound search. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pages 479–488, Corvallis, Oregon. AUAI Press.
- Andrew Moore and Weng-Keen Wong. 2003. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In *Intl. Conf. on Machine Learning*, pages 552–559.
- S. Ott, S. Imoto, and S. Miyano. 2004. Finding optimal models for small gene networks. In *Pac. Symp. Biocomput*, pages 557–567.
- Pekka Parviainen and Mikko Koivisto. 2009. Exact structure discovery in Bayesian networks with less space. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, Montreal, Quebec, Canada. AUAI Press.
- Judea Pearl. 1984. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Ira Pohl. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(3-4):193 – 204.
- Tomi Silander and Petri Myllymaki. 2006. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, Arlington, Virginia. AUAI Press.
- Ajit Singh and Andrew Moore. 2005. Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, June.
- Yoshinori Tamada, Seiya Imoto, and Satoru Miyano. 2011. Parallel algorithm for learening optimal bayesian network structure. *Journal of Machine Learning Research*, 12:2437–2459.
- Marc Teyssier and Daphne Koller. 2005. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 584–590, Arlington, Virginia. AUAI Press.
- Richard Valenzano, Nathan Sturtevant, Jonathan Schaeffer, Karen Buro, and Akihiro Kishimoto. 2010. Simultaneously searching with multiple settings: An alternative to parameter tuning for suboptimal single-agent search algorithms. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS 2010)*.
- Changhe Yuan and Brandon Malone. 2012. An improved admissible heuristic for finding optimal bayesian networks. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence (UAI-12)*. AUAI Press.
- Changhe Yuan, Brandon Malone, and Xiojian Wu. 2011. Learning optimal Bayesian networks using A\* search. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*.
- Weixiong Zhang and Joseph C. Pemberton. 1994. Epsilon-transformation: exploiting phase transitions to solve combinatorial optimization problemsinitial results. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, AAAI'94, pages 895–900, Menlo Park, CA, USA. American Association for Artificial Intelligence.

# An interactive approach for cleaning noisy observations in Bayesian networks with the help of an expert

Andrés R. Masegosa and Serafín Moral

{andrew,smc}@decsai.ugr.es

Department of Computer Science and Artificial Intelligence  
University of Granada

## Abstract

When using Bayesian networks in real applications it is often the case that the empirical evidence or observations we employ for making inferences are corrupt and contain noise: Failure in a sensor, outliers, human errors, etc. Although many methods have been proposed in the literature for data cleaning (i.e. detect and correct noisy data values), all of these methods perform this task automatically. In this paper we argue that, if available, expert knowledge should be used for this task and we propose two methods which explicitly interact with an expert for detecting and correcting noisy observations.

## 1 Introduction

A Bayesian network (BN) is a statistical model (Pearl, 1988) that graphically encodes, via a directed acyclic graph, the conditional independencies among the domain variables. The applications of BNs in real problems are very well known and diverse (Pourret et al., 2008): Medicine, recognition, gambling, monitoring, forensic reasoning, genomics, etc.

One of the key properties of these models is the possibility of introducing empirical evidence about any of the variables (e.g. the result of a medical test) and, by means of any of the available evidence propagation algorithms (Jensen and Nielsen, 2007), updating the beliefs about the distribution over the rest of the variables (e.g. the causes of a given disease). However, in real applications, it may happen that the empirical observations or evidence introduced in the model are corrupt and, in consequence, the results of the reasoning provided by these systems may be corrupt too. For example, in a medical diagnosis system relying in a BN model, a doctor may wrongly introduce the result of a test by pressing an incorrect keyboard key and this could cause a wrong disease diagnosis. But the corruption of empirical evidence may be due to many other causes: Failure in a sensor; noise

in a transmission channel, outliers, etc. There are many problem domains where the empirical evidence is subject to some kind of corruption process (Zhu et al., 2007).

One of the most studied corruption processes is the presence of noisy values in the empirical observations or data used to make inferences (Zhu and Wu, 2006). For data mining problems, there is a large amount of literature on automatic data cleaning methods (Maletic and Marcus, 2010), and some of these methods were specifically designed for BNs (Doshi et al., 2003).

In this work, we explore an alternative approach to the problem of detecting and correcting noisy observations in multinomial data. We propose new methods which have two main properties. First, we explicitly model the noisy process which determines the value of the noisy observable variables  $O'_i$  as a function of the true value of these observations  $O_i$ . Secondly, we assume that an expert will be available to clean up noisy observations. Thus, in contrast with the previous approaches, based on automatic methods in the sense that they do not allow human intervention, we propose a method which explicitly interacts with an expert. So, given a BN and an observation vector of some of these vari-

ables (i.e. the variables that can be observed, such as a medical test or a symptom), and given that some of the values can be noisy, our method will try to detect these noisy values and correct them to their actual values. For this purpose, our method will interact with an expert by requesting information about some particular values of evidence (i.e. the expert will have to confirm if an observed value is noisy or not and, if noisy, provide the actual value). Our method will try to ease and minimize this interaction. In the field of natural language processing, similar approaches are quite popular to correct misspelled words when typing on a mobile phone. Instead of correcting automatically a misspelled word, these systems display a set of alternative correct words and let the user decide which is the correct one.

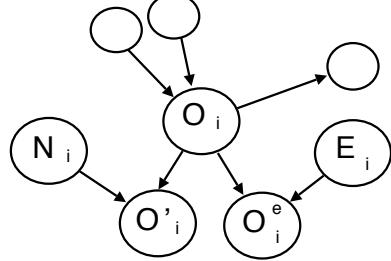
The remainder of this paper is organized as follows. Section 2 details the noisy and the expert model considered in this approach. Section 3 introduces our two proposed methods for cleaning noisy observations with the help of an expert. In Section 4 we experimentally evaluate our approach and, finally, Section 5 outlines the main conclusions and plans for future work.

## 2 Modelling noisy observations and expert knowledge

Let us assume that we have a problem domain which is defined by two vectors of multinomial random variables: A vector of explanatory variables  $\mathbf{X} = \{X_1, \dots, X_n\}$  (variables for which we want to make inferences, e.g., the cause of given disease) and a vector of observable variables  $\mathbf{O} = \{O_1, \dots, O_p\}$  (e.g. the result of a test, the measure of a given sensor, etc.) Let us also assume that the joint probability distribution of these variables in our problem domain,  $P(\mathbf{X}, \mathbf{O})$ , is modelled using a known Bayesian network (BN), denoted by  $\mathcal{B}$ .

As mentioned in the introduction, we consider that when gathering observations from the observable variables  $\mathbf{O}$  in our problem domain, there are mechanisms which add noise and corrupt some of the observed values. To account for these corruption processes, we introduce new

Figure 1: A expanded BN modelling noisy observations and expert knowledge



variables in the BN.  $\mathbf{O}' = \{O'_1, \dots, O'_p\}$  denotes a vector of noisy observable variables where  $O'_i$  represents a noisy observable variable which can be equal to  $O_i$  or not, depending on whether a noisy event (which corrupts the observed value) takes place or not when observing this variable. We assume that the set of possible values of  $O'_i$ ,  $\Omega(O'_i)$ , is identical to the set of possible values  $\Omega(O_i)$  of  $O_i$ . The noisy events are also explicitly modelled by a vector of variables denoted by  $\mathbf{N} = \{N_1, \dots, N_p\}$  where  $N_i$  models the occurrence (or not) of a noisy event when observing the value associated to variable  $O_i$ . Thus, each  $N_i$  has two values:  $\Omega(N_i) = \{Noise, NoNoise\}$ . These new sets of variables are related to each other as shown in Figure 1, i.e. we assume that the noisy event variables are independent and that each observed variable  $O'_i$  only depends on the true value  $O_i$  and the noisy event  $N_i$ .

We use the notation  $P(N_i = Noise) = \tau_i$ , where  $\tau_i \in [0, 1]$  but usually is a small probability value. The conditional probability  $P(O'_i|O_i, N_i)$  defines the noise model and it will vary depending on the particular problem we are dealing with. In this work we assume a simple noise model defined as follows:  $P(O'_i = o_i|O_i = o_i, N_i = noNoise) = 1$ , i.e. the identity function;  $P(O'_i = o_i|O_i \neq o_i, N_i = Noise) = \frac{1}{|\Omega(O_i)| - 1}$  where  $|\cdot|$  is the cardinality operator, i.e. when noise is present, the conditional is a uniform distribution over the rest of values which differ from the current value of  $O_i$ .

Similarly, we also model the possibility of introducing expert knowledge, as discussed in the introduction, by adding two new sets of variables  $\mathbf{O}^e = \{O^e_1, \dots, O^e_p\}$  and  $\mathbf{E} = \{E_1, \dots, E_p\}$ .

Variable  $O_i^e$  models the knowledge provided by the expert about the real value of variable  $O_i$  and variable  $E_i$  models the correctness of the expert's knowledge (i.e. the expert may be wrong and provide incorrect knowledge):  $\Omega(E_i) = \{Right, Wrong\}$ . Figure 1 shows how these variables are connected in the BN used to model the interaction procedure.  $P(E_i = Wrong) = \eta_i$ , where  $\eta_i \in [0, 1]$  and usually is a small probability value. The conditional probability  $P(O_i^e|O_i, E_i)$  is defined as follows:  $P(O_i^e = o_i|O_i = o_i, E_i = Correct) = 1$ , i.e. the identity function;  $P(O_i^e = o_i|O_i \neq o_i, E_i = Wrong) = \frac{1}{|\Omega(O_i)|-1}$ , i.e. the conditional when the expert is wrong is a uniform distribution over the rest of values which differ from the current value of  $O_i$ .

### 3 Cleaning noisy observations with the help of an expert

Our problem starts with a vector of noisy observations, denoted by  $\mathbf{o}' = \{o'_1, \dots, o'_p\} \in \Omega(\mathbf{O}')$  and our goal is to recover the vector of actual observations, denoted by  $\mathbf{o} \in \Omega(\mathbf{O})$ .

In this work we explore two different strategies to approach the problem of deciding which knowledge should be requested to an expert.

#### 3.1 An entropy-based approach

Here we follow a pure inference approach where we do not consider the existence of any specific cost or utility when cleaning noisy observations. With this approach, the output is the most probable explanation (Gamez, 2003), denoted by  $\mathbf{o}^{MPE}$ , of the observable variables given the noisy observation,  $\mathbf{o}'$ , and the information provided by the expert:

$$\mathbf{o}^{MPE} = \arg \max_{\mathbf{O}=\mathbf{o}} P(\mathbf{O} = \mathbf{o} | \mathbf{O}' = \mathbf{o}', \mathbf{o}^e)$$

where  $\mathbf{o}^e$  denotes the knowledge provided by the expert which is introduced as evidence for some of the variables in  $\mathbf{O}^e$  (see Figure 1).

The introduction of expert knowledge is used to discard alternative explanations or assignments which have a non-negligible probability,  $P(\mathbf{O} = \mathbf{o} | \mathbf{O}' = \mathbf{o}') > 0$  with  $\mathbf{o} \neq \mathbf{o}^{MPE}$ ,

and improve the confidence in the retrieved explanation  $\mathbf{o}^{MPE}$ . To quantify this confidence we employ the conditional entropy of the observable variables:  $H(\mathbf{O} | \mathbf{O}' = \mathbf{o}') = \sum_{\mathbf{o} \in \Omega(\mathbf{O})} P(\mathbf{o} | \mathbf{o}') \ln P(\mathbf{o} | \mathbf{o}')$ . A null conditional entropy means that  $\mathbf{o}^{MPE}$  accumulates all the mass probability, while a maximum conditional entropy means that all explanations are equally likely. So, our approach is based on requesting knowledge about the real value of the variable  $O_i$  which most reduces this entropy measure. That is to say, we iteratively ask the expert for her/his belief about the value of the  $O_{max}^e$  variable with the highest information gain or mutual information about the observable variables  $\mathbf{O}$ :

$$O_{max}^e = \arg \max_{O_i^e} IG(\mathbf{O}; O_i^e | \mathbf{o}', \mathbf{o}^e)$$

where  $\mathbf{o}^e$  refers to the set of answers (i.e. evidence for some of the  $O_i^e$  variables) given by the expert so far (at the beginning, this set is empty, i.e. there is no expert knowledge).

The computation of  $IG(\mathbf{O}; O_i^e | \mathbf{o}', \mathbf{o}^e)$  seems to be complex, as  $\mathbf{O}$  is a multidimensional variable, but it can be easily computed if we take into account that  $O_i^e$  is independent of  $\mathbf{O} - \{O_i\}$  given  $O_i$ . So each variable  $O_i^e$  only gives information about its associated variable,  $O_i$ :

$$IG(\mathbf{O}; O_i^e | \mathbf{o}', \mathbf{o}^e) = H(O_i^e | \mathbf{o}', \mathbf{o}^e) - \sum_{\mathbf{o} \in \Omega(\mathbf{O})} P(\mathbf{o} | \mathbf{o}', \mathbf{o}^e) H(O_i^e | \mathbf{o}', \mathbf{o}^e, \mathbf{o})$$

and  $\sum_{\mathbf{o} \in \Omega(\mathbf{O})} P(\mathbf{o} | \mathbf{o}', \mathbf{o}^e) H(O_i^e | \mathbf{o}', \mathbf{o}^e, \mathbf{o}) = \sum_{o_i \in \Omega(O_i)} P(o_i | \mathbf{o}', \mathbf{o}^e) H(O_i^e | o_i)$ . Taking into account the probabilities relating  $O_i$  with  $O_i^e$ , we have that  $H(O_i^e | o_i) = -\tau_i \ln \tau_i - (1 - \tau_i) \ln \frac{1 - \tau_i}{|\Omega(O_i)| - 1} = H(E_i) + (1 - \tau_i) \ln(|\Omega(O_i)| - 1)$ . Putting everything together, we get that  $IG(\mathbf{O}; O_i^e | \mathbf{o}', \mathbf{o}^e)$  is equal to:

$$H(O_i^e | \mathbf{o}', \mathbf{o}^e) - H(E_i) + (1 - \tau_i) \ln(|\Omega(O_i)| - 1)$$

which can be computed by standard propagation algorithms.

We stop requesting knowledge when the information given by  $O_{max}^e$  is below a given threshold, denoted by  $\lambda$ . This parameter defines the information gain we are willing to accept in exchange for a question to the expert. A pseudo-code description is given in Algorithm 1.

### Algorithm 1. The entropy based method

```

1: $\mathbf{o}^e = \emptyset$.
2: end=false;
3: repeat
4: Compute the O_i^e variable with the highest information gain:

$$O_{max}^e = \arg \max_{O_i^e} IG(\mathbf{O}; O_i^e | \mathbf{o}', \mathbf{o}^e)$$

5: if $IG(\mathbf{O}; O_i^e | \mathbf{o}', \mathbf{o}^e) > \lambda$ then
6: Ask to expert about O_{max}^e and introduce expert's answer: $\mathbf{o}^e = \mathbf{o}^e \cup o_{max}^e$.
7: else
8: end=true;
9: end if
10: until end
11: $\mathbf{o}^{MPE} = \arg \max_{\mathbf{o}=\mathbf{o}} P(\mathbf{O} = \mathbf{o} | \mathbf{O}' = \mathbf{o}', \mathbf{o}^e)$;
12: return \mathbf{o}^{MPE} ;

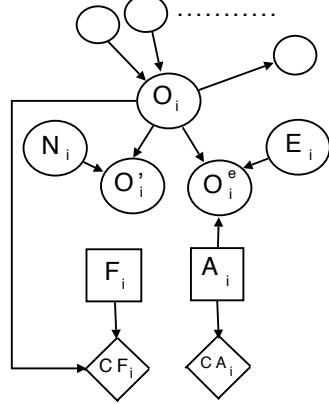
```

This algorithm converges and stops the querying process for a positive threshold because the information gain is by definition lower than the entropy of the observable variables  $\mathbf{O}$  and this entropy is always reduced with each new answer from the expert. The algorithm also involves the computation of the most probable explanation (line 11). Although this task can be intractable for some networks, many efficient approximate methods have been proposed in the literature (Jensen and Nielsen, 2007).

### 3.2 A cost-based approach

In this new approach we explicitly consider that there are costs or utilities associated with this problem. We assume that there is a specific and known cost, denoted by  $CF$ , which depends on the differences between the actual values of the observed variables  $\mathbf{o}$  and the values assigned for the system, and that there is also a specific and known cost, denoted by  $CA_i$ , when we ask the expert about the noisy observable variable  $O_i$ . So, this problem can be approached as a decision making problem where we are going to have observations in variables  $\mathbf{O}'$  and, based on these

Figure 2: The Influence Diagram employed in the cost-based approach of Section 3.2



observations, we have to make the following decisions: If we request expert knowledge about a noisy observable variable  $O'_i$ , this decision is denoted by  $A_i$  and has two states,  $\{\text{Ask}, \text{NotAsk}\}$ ; and the decision regarding the necessity of correcting this noisy variable,  $O'_i$ , and the way of doing, is denoted by  $F_i$  and has the same set of states of  $O_i$  (i.e. when  $F_i = O'_i$ , we are deciding that there is no noise in this observation, otherwise  $F_i$  takes the corrected value). When the  $F_i$  decisions are taken, all the noisy observable variables are fixed.

Although the above decision making problem can be represented by a decision graph or an influence diagram, solving this problem (i.e. finding an optimal policy) is not computationally feasible if the number of observable variables  $p$  is relatively large, for two reasons: There is no sequential order in the decisions (except that decisions about fixing are made after decisions about asking), i.e. we have an unconstrained influence diagram (Jensen and Nielsen, 2007) with exponential complexity in time; and each decision depends on the information of  $2 \cdot p$  variables (i.e.  $\mathbf{O}' \cup \mathbf{O}^e$ ). Thus, the size of the decision node tables is exponential in  $p$ .

In order to make feasible the solution of this decision problem for a relatively large number of noisy observable variables, we propose the following simplifications: (i) the decision making problem is defined and solved only for a partic-

ular noisy observation vector  $\mathbf{o}'$  (i.e. we assume that there are no alternative noisy observations besides the current one); (ii) the overall cost of incorrectly fixing a noisy observation vector,  $\mathbf{o}'$ , can be computed as a sum of independent costs for each variable,  $CF = \sum_{i=1}^p CF_i$ ; (iii) we assume that we have  $p$  different decision problems. Each of them,  $\mathcal{D}_i$ , is the problem obtained by considering only decision  $A_i$  and all the fixing-related decision variables  $\{F_1, \dots, F_p\}$ . When solving  $\mathcal{D}_i$  we assume that we do not ask for the rest of the variables:  $A_j = \text{NotAsk}$  ( $j \neq i$ ). To implement this last simplification we initially set the evidence in the variables  $O_j^e$  with  $j \neq i$  to value “NA”, which means that we do not have yet any answer from the expert regarding variable  $O_j^e$ . Thus, we extend the set values of  $O_i^e$ ,  $\Omega(O_i^e) = \Omega(O_i) \cup \{\text{NA}\}$ , and redefine the conditional probability of  $O_i^e$ , which now depends on the decision  $A_i$ . This conditional probability is equally defined as detailed in Section 2 when  $A_i = \text{Ask}$ ,  $P(O_i^e = \text{NA}|O_i, E_i, A_i = \text{Ask}) = 0$ , but when  $A_i = \text{NotAsk}$  we define that  $P(O_i^e = \text{NA}|O_i, E_i, A_i = \text{NotAsk}) = 1$ .

Under these simplifications, our decision making problem can be represented by means of the unconstrained influence diagram depicted in Figure 2, which extends our previous BN model detailed in Section 2. This problem is then solved by greedily selecting the sequence of variables  $O_i$  for which we ask the expert (i.e. the order in which the decision problems  $\mathcal{D}_i$  are solved) and, after that, by choosing the optimal set of fixing decisions  $F_i$ . Each time we decide to ask the expert about a variable  $O_i$ , the obtained value  $O_i^e = o_i^e$  is added to the current set of observations  $\mathbf{O}^e = \mathbf{o}^e$ . The following decision about asking is made after conditioning each one of the original problems  $\mathcal{D}_i$  to  $\mathbf{O}^e = \mathbf{o}^e$ ,  $\mathbf{O}' = \mathbf{o}'$  and only for these concrete values. If we have already asked the expert about variable  $O_i$ , then in subsequent problems it is not necessary to solve  $\mathcal{D}_i$ , and in all problems  $\mathcal{D}_j$  ( $j \neq i$ ) we assume that  $A_i = \text{Ask}$ .

When solving a problem  $\mathcal{D}_i$  we evaluate the influence diagram for  $A_i = \text{Ask}$  (problem  $\mathcal{D}_i^a$ ) and for  $A_i = \text{NotAsk}$  (problem  $\mathcal{D}_i^n$ ). To evaluate these two problems we fix the decision vari-

able to the corresponding value, propagate the information, and optimize the decision variables about fixing  $(F_1, \dots, F_q)$ . In that step we have to take into account that the optimal values of these variables do not depend on the order in which we decide to fix them, as each time we fix a variable  $F_j = f_j^*$ , this value does not change the probabilities of any other chance variable in the problem. Also, the selection can be independently computed for any of the variables  $F_j$ . In problem  $\mathcal{D}_i^n$ , we must compute for each variable  $F_i$  the value  $f_j^*$  such that

$$f_j^* = \arg \min_{f_j} \sum_{o_j} CF_j(f_j, o_j) P(o_j | \mathbf{o}', \mathbf{o}^e)$$

where the above minimization problem can be solved in constant time,  $O(|\Omega(O_i)|)$  after the observations have been propagated. For example, if we have a 0/1 cost error (i.e.  $CF(f_i, o_i) = 1$  if  $f_i \neq o_i$  and  $CF(f_i, o_i) = 0$  if  $f_i = o_i$ ), the above problem reduces to selecting the  $o_i$  with the highest probability.

Let us denote by  $CF_j(\mathbf{o}', \mathbf{o}^e)$  the value  $\sum_{o_j} CF_j(f_j^*, o_j) P(o_j | \mathbf{o}', \mathbf{o}^e)$ . The cost of problem  $\mathcal{D}_i^n$ ,  $c(\mathcal{D}_i^n)$ , will be equal to  $\sum_j CF_j(\mathbf{o}', \mathbf{o}^e)$ .

The cost of problem  $\mathcal{D}_i^a$  can be computed in a similar way. The only differences are that now we have to add the cost of asking for a variable  $CA_i$  and that each time we fix the value of variable  $O_j$  (we compute  $f_j^*$ ), we will have an additional observation  $O_i^e = o_i^e$ . If we denote by  $CF_j(\mathbf{o}', \mathbf{o}^e \cup o_i^e)$  the optimal cost of fixing variable  $O_j$  with these observations, the cost of the problem will be  $c(\mathcal{D}_i^a) = \sum_{o_i^e} P(o_i^e | \mathbf{o}', \mathbf{o}^e) (\sum_j CF_j(\mathbf{o}', \mathbf{o}^e \cup o_i^e)) + CA_i$ .

To select the variable  $O_i$  for which we are going to ask the expert, we compute the differences  $c(\mathcal{D}_i^a) - c(\mathcal{D}_i^n)$ . This value is denoted by  $CG(A_i | \mathbf{o}', \mathbf{o}^e)$  and called the expected cost gain if we ask for variable  $O_i$ . We decide to ask for the variable  $O_i$  with a greatest cost gain  $CG(A_i | \mathbf{o}', \mathbf{o}^e)$ . The observation provided by the expert  $O_i^e = o_i^e$  is added to  $\mathbf{o}^e$  and the process is repeated. The stop condition is that this greater cost gain value is lower or equal than 0 (we experiment a loss when asking for any of the variables). So, this method resembles the

one detailed in Algorithm 1 but using the cost gain instead of the information gain and, finally, retrieving  $\mathbf{f}^*$  instead of the most probable explanation.

Note that value  $c(\mathcal{D}_i^a)$  is the same for all the decision problems  $\mathcal{D}_i$  and should be computed only once.

### 3.3 Using the EM algorithm to estimate the noise rate

In the two previously described methods we assumed that the Bayesian network  $\mathcal{B}$  relating explanatory and observable variables,  $(\mathbf{X}, \mathbf{O})$  and the conditional probabilities of variables  $\mathbf{O}'$  are known. For this procedure we are going to assume that the prior probabilities for noisy observations  $P(N_i = \text{Noise}) = \tau_i$  are not known; instead, we are given a pool of noisy observations,  $D = \{\mathbf{o}'_{(1)}, \dots, \mathbf{o}'_{(M)}\}$  where we can estimate them. In this section we describe how these unknown parameters  $\tau = (\tau_1, \dots, \tau_p)$  can be estimated by means of the EM algorithm (Dempster et al., 1977) (once estimated, the previously detailed cleaning approaches can be used for data cleaning as usual). Using this algorithm, we get the maximum a posteriori (MAP) estimate of the parameters:  $\hat{\tau} = \max_{\tau} L(D; \tau)$ , where  $L(D; \tau)$  is a function on  $\tau$  proportional to the posterior probability which is computed as follows:

$$L(D; \tau) = P(\tau) \prod_m \sum_{\mathbf{N}, \mathbf{O}} P(\mathbf{o}'_{(m)}, \mathbf{N}, \mathbf{O} | \tau)$$

where  $P(\tau)$  is the prior over the parameters. In this work, it is defined as follows:  $P(\tau) = \prod_i P(\tau_i) = \prod_i B(\tau_i; 0.1, 1.9)$ , where  $B(\cdot; \cdot)$  is the Beta distribution. In this Beta distribution we assume that the prior belief about the parameter  $\tau$  (the error rate) is  $1/20 = 0.05$ .

The EM algorithm is an iterative method which converges to the MAP estimate,  $\hat{\tau}$ , by alternating the two following steps:

An **Expectation (E) step**, which computes the expectation of the log-likelihood evaluated using the current estimate for the parameters in step  $k$ ,  $\tau^{(k)}$ . This is implemented here by

computing  $P(N_i = \text{noise} | \mathbf{o}'_{(m)}, \tau^{<k>})$  for each of the noisy observation vectors in  $D$ .

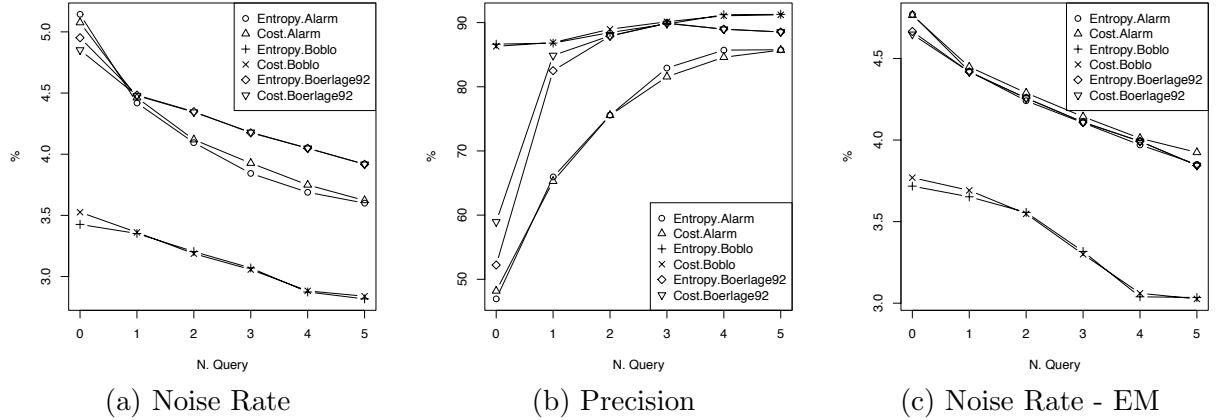
A **Maximization (M) step**, which updates the parameters maximizing the expected log-likelihood found on step E. This is implemented in this problem as follows:  $\tau_i^{<k+1>} = \frac{1}{M} \sum_{m=1}^M P(N_i = \text{noise} | \mathbf{o}'_{(m)}; \tau^{<k>})$ .

The algorithm is run until convergence:  $\sum_i |\tau_i^{<k+1>} - \tau_i^{<k>}| < 1e - 5$ .

## 4 Experimental evaluation

In this experimental evaluation we assess the techniques presented in Section 3. For that purpose, we consider three different BNs (the Alarm network with 37 nodes, the Boblo network with 23 nodes, and the Boerlage network, with 23 nodes too) and, by means of logic sampling, we artificially generate different random samples  $(\mathbf{o}, \mathbf{o}', \mathbf{n}, \mathbf{o}^e)$  of the variables in  $\mathbf{O}$ ,  $\mathbf{O}'$ ,  $\mathbf{N}$  and  $\mathbf{O}^e$ , extending previously each model according to Figure 1. For each noisy observation generated,  $\mathbf{o}'$ , we then apply the two methods detailed in Section 3.1 and 3.2, in order to evaluate their capacity to detect and correct the noisy observations (i.e. those  $\mathbf{o}'_i$  such that  $n_i = \text{Noise}$ ). The presence of expert knowledge is simulated by accessing the  $\mathbf{o}^e_i$  values, considering also the possibility that the expert gives a wrong answer. Finally, once the cleaning method has finished we can check how well this method works by comparing the cleaned observations with the actual ones,  $\mathbf{o}$ . Two different error measures are considered to evaluate the performance of the methods: the *a posteriori noise rate* as the rate of values in  $\mathbf{o}'$  which are still noisy (i.e.  $\mathbf{o}'_i^{MPE} \neq o_i$  in the entropy-based approach or  $f_i^* \neq o_i$  in the cost-based approach) after the interaction with the expert; and the *precision of the fixing decisions* as the ratio between the number of correctly identified noisy values (those  $\mathbf{o}'_i$  such that  $n_i = \text{Noise}$  and which are correctly cleaned) over the total number of cleaned observations (i.e. those values in  $\mathbf{o}'$  which are either correctly or incorrectly cleaned because they have been identified as noisy observations by our methods). With this second measure we aim to evaluate if our meth-

Figure 3: Evaluation of the entropy and the cost based methods for different fixed number of queries and for data samples where 5% of the observations are noisy (i.e. when the noise rate is below this value, it means that some noisy observations have been correctly fixed).



ods fix only those observations which are really noisy, and do not incorrectly fix those which are not noisy.

In this evaluation we consider all the variables in the three BNs mentioned above as observable variables (i.e.  $\mathbf{X} = \emptyset$ ). We also set a noise rate equal to 5% for all the variables (i.e.  $\forall i \tau_i = 0.05$ ). We have evaluated different noise rates and the conclusions are quite similar. The reliability of the expert is set to 99% (i.e.  $\forall i \eta_i = 0.01$ ). For the cost-based approach, we have considered a standard 0/1 cost error.

The results of this evaluation are displayed in Figures 3 and the displayed error measures of the two cleaning methods (labeled as *Entropy-<BN>* and *Cost-<BN>*) are computed as an average value over 1000 observation vectors sampled for the three BNs. Although the two proposed methods consider different stop criteria which depend on the particular problem, we have only evaluated the performance of the methods using a preset number of queries which is displayed in the X-axis of the figures. As can be seen, we have considered from 1 to 5 queries, but also the case where there are no queries submitted to the expert,  $N.Query=0$ . In this last case, we have a fully automatic method without expert intervention.

In Figure 3 (a), we can firstly see that for Boerlage92 and Alarm networks there is hardly

any improvement in the noise rate using the methods that do not require help from the expert ( $N.Query = 0$ ), i.e. simply cleaning the noisy observations using the most probable explanation in the case of the entropy based method (see Section 3.1) or using the fixing decisions with the lowest expected cost in the case of the cost-based approach (see Section 3.2). Actually, in the case of the Alarm network there is a slight increment in the noise rate (5.14% for the entropy method and 5.08% for the cost method). This is due, as can be seen in Figure 3 (b), to the *precision* of the fixing decisions, which is very low at  $N.Query=0$ . This contrasts with the Boblo network, where the precision is high and the noise rate is reduced. In conclusion, we can see that, although there are models where automatic cleaning methods work, there are other models where it can be very hard to successfully clean the noisy observations without the help of an expert, even having perfect knowledge of the underlying probability distribution.

In any case, we can see that the noise rates of the observations decrease with our methods, as more queries are submitted to the expert. The strength and the pace of this decrease strongly depend on the particular model. The same trend appears for the precision of noisy values detection. The introduction of expert knowl-

edge increases the precision of noisy observations detection. This is also remarkable because there is a significant decrease in the number of false positive detections (i.e. observations incorrectly considered as noisy).

On the other hand, we can see that the entropy-based and the cost-based methods perform quite similarly in these models, although we stress that the latter method can be tailored to situations where there are asymmetric costs when cleaning noisy observations. It also allows to explicitly consider the cost of requesting knowledge to an expert and, thus, it clearly states when to stop asking questions.

Finally, in Figure 3 (c), we show the results of applying our methods, but this time using the EM algorithm as a previous step (see Section 3.3) to estimate the noise rates of the noisy observable variables. These estimates are obtained using the same pool of 1000 samples used in the previous evaluation. As can be seen in this figure, the overall result is quite similar to the case where the noise rates are known. We cannot give details about the estimations of noise rates  $\tau_i$  obtained by using EM due to lack of space, but they are accurate and tend to be below the true value. This happens because some noisy observations cannot be identified and the EM considers them as not noisy. We then show that a large set of noisy observations can be employed to obtain estimates of the noise rates and then apply successfully our proposed methods to integrate the expert knowledge and fix many of the noisy observations.

## 5 Conclusions and future works

In this work we have presented two approaches for cleaning noisy observations using expert knowledge. As opposed to fully automatic methods, our proposals interact with an expert and request knowledge about particular noisy observations. Our empirical evaluation shows that the inclusion of expert knowledge correctly fixes many noisy values. We also show that, in some cases, this can not be achieved without the help of an expert.

Future work will be focused on the exten-

sion of these methods to scenarios where the Bayesian network that models the joint probability distribution is not known and must be learned from a pool of noisy data samples.

## Acknowledgements

This work has been jointly supported by the research programme Consolider Ingenio 2010, the Spanish Ministerio de Ciencia de Innovación and the Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía under projects CSD2007-00018, TIN2010-20900-C04-01, TIC-6016 and P08-TIC-03717, respectively. We also thank the reviewers for their constructive suggestions.

## References

- A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.
- Prashant Doshi, Lloyd Greenwald, and John R. Clarke. 2003. Using Bayesian networks for cleansing trauma data. In Ingrid Russell and Susan M. Haller, editors, *FLAIRS Conference*, pages 72–76. AAAI Press.
- J.A. Gamez. 2003. Abductive inference in Bayesian networks: A review. Technical report, Department of Computer Science, University of Castilla-La Mancha, UCLM.
- Finn V. Jensen and Thomas D. Nielsen. 2007. *Bayesian Networks and Decision Graphs*. Information Science and Statistics. Springer-Verlag, New York, USA.
- Jonathan I. Maletic and Andrian Marcus. 2010. Data cleansing: A prelude to knowledge discovery. pages 19–32.
- J. Pearl. 1988. *Probabilistic Reasoning with Intelligent Systems*. Morgan & Kaufman, San Mateo.
- Olivier Pourret, Bruce Marcot, and Patrick Naim. 2008. *Bayesian networks: a practical guide to applications*. Statistics in Practice. Wiley, Chichester.
- Xingquan Zhu and Xindong Wu. 2006. Error awareness data mining. In *Granular Computing, 2006 IEEE International Conference on*, pages 269 – 274, may.
- Xingquan Zhu, Taghi M. Khoshgoftaar, Ian Davidson, and Shichao Zhang. 2007. Editorial: Special issue on mining low-quality data. *Knowl. Inf. Syst.*, 11(2):131–136, February.

# Learning AMP Chain Graphs under Faithfulness

Jose M. Peña

ADIT, IDA, Linköping University, SE-58183 Linköping, Sweden

jose.m.pena@liu.se

## Abstract

This paper deals with chain graphs under the alternative Andersson-Madigan-Perlman (AMP) interpretation. In particular, we present a constraint based algorithm for learning an AMP chain graph a given probability distribution is faithful to. We also show that the extension of Meek's conjecture to AMP chain graphs does not hold, which compromises the development of efficient and correct score+search learning algorithms under assumptions weaker than faithfulness.

## 1 Introduction

This paper deals with chain graphs (CGs) under the alternative Andersson-Madigan-Perlman (AMP) interpretation (Andersson et al., 2001). In particular, we present an algorithm for learning an AMP CG a given probability distribution is faithful to. To our knowledge, we are the first to present such an algorithm. However, it is worth mentioning that, under the classical Lauritzen-Wermuth-Frydenberg (LWF) interpretation of CGs (Lauritzen, 1996), such an algorithm already exists (Ma et al., 2008; Studený, 1997). Moreover, we have recently developed an algorithm for learning LWF CGs under the milder composition property assumption (Peña et al., 2012).

The AMP and LWF interpretations of CGs are sometimes considered as competing and, thus, their relative merits have been pointed out (Andersson et al., 2001; Drton and Eichler, 2006; Levitz et al., 2001; Roverato and Studený, 2006). Note, however, that no interpretation subsumes the other: There are many independence models that can be induced by a CG under one interpretation but that cannot be induced by any CG under the other interpretation (Andersson et al., 2001, Theorem 6).

The rest of the paper is organized as follows. Section 2 reviews some concepts. Section 3 presents the algorithm. Section 4 proves its correctness. Section 5 closes with some discussion.

## 2 Preliminaries

In this section, we review some concepts from probabilistic graphical models that are used later in this paper. All the graphs and probability distributions in this paper are defined over a finite set  $V$ . All the graphs in this paper are hybrid graphs, i.e. they have (possibly) both directed and undirected edges. The elements of  $V$  are not distinguished from singletons. We denote by  $|X|$  the cardinality of  $X \subseteq V$ .

If a graph  $G$  contains an undirected (resp. directed) edge between two nodes  $V_1$  and  $V_2$ , then we write that  $V_1 - V_2$  (resp.  $V_1 \rightarrow V_2$ ) is in  $G$ . The parents of a set of nodes  $X$  of  $G$  is the set  $pa_G(X) = \{V_1 | V_1 \rightarrow V_2 \text{ is in } G, V_1 \notin X \text{ and } V_2 \in X\}$ . The neighbors of a set of nodes  $X$  of  $G$  is the set  $ne_G(X) = \{V_1 | V_1 - V_2 \text{ is in } G, V_1 \notin X \text{ and } V_2 \in X\}$ . The adjacents of a set of nodes  $X$  of  $G$  is the set  $ad_G(X) = \{V_1 | V_1 \rightarrow V_2, V_1 - V_2 \text{ or } V_1 \leftarrow V_2 \text{ is in } G, V_1 \notin X \text{ and } V_2 \in X\}$ . A route from a node  $V_1$  to a node  $V_n$  in  $G$  is a sequence of (not necessarily distinct) nodes  $V_1, \dots, V_n$  such that  $V_i \in ad_G(V_{i+1})$  for all  $1 \leq i < n$ . The length of a route is the number of (not necessarily distinct) edges in the route, e.g. the length of the route  $V_1, \dots, V_n$  is  $n - 1$ . A route is called a cycle if  $V_n = V_1$ . A route is called descending if  $V_i \in pa_G(V_{i+1}) \cup ne_G(V_{i+1})$  for all  $1 \leq i < n$ . The descendants of a set of nodes  $X$  of  $G$  is the set  $de_G(X) = \{V_n | \text{there is a descending route from } V_1 \text{ to } V_n \text{ in } G, V_1 \in X \text{ and } V_n \notin X\}$ . A cycle

is called a semidirected cycle if it is descending and  $V_i \rightarrow V_{i+1}$  is in  $G$  for some  $1 \leq i < n$ . A chain graph (CG) is a hybrid graph with no semidirected cycles. A set of nodes of a CG is connected if there exists a route in the CG between every pair of nodes in the set st all the edges in the route are undirected. A connectivity component of a CG is a connected set that is maximal wrt set inclusion. The connectivity component a node  $A$  of a CG  $G$  belongs to is denoted as  $co_G(A)$ . The subgraph of  $G$  induced by a set of its nodes  $X$  is the graph over  $X$  that has all and only the edges in  $G$  whose both ends are in  $X$ . An immorality is an induced subgraph of the form  $A \rightarrow B \leftarrow C$ . A flag is an induced subgraph of the form  $A \rightarrow B - C$ . If  $G$  has an induced subgraph of the form  $A \rightarrow B \leftarrow C$  or  $A \rightarrow B - C$ , then we say that the triplex  $(\{A, C\}, B)$  is in  $G$ . Two CGs are triplex equivalent iff they have the same adjacencies and the same triplexes.

A node  $B$  in a route  $\rho$  is called a head-not-tail node in  $\rho$  if  $A \rightarrow B \leftarrow C$ ,  $A \rightarrow B - C$ , or  $A - B \leftarrow C$  is a subroute of  $\rho$  (note that maybe  $A = C$  in the first case). Let  $X$ ,  $Y$  and  $Z$  denote three disjoint subsets of  $V$ . A route  $\rho$  in a CG  $G$  is said to be  $Z$ -open when (i) every head-not-tail node in  $\rho$  is in  $Z$ , and (ii) every other node in  $\rho$  is not in  $Z$ . When there is no route in  $G$  between a node in  $X$  and a node in  $Y$  that is  $Z$ -open, we say that  $X$  is separated from  $Y$  given  $Z$  in  $G$  and denote it as  $X \perp_G Y|Z$ .<sup>1</sup> We denote by  $X \not\perp_G Y|Z$  that  $X \perp_G Y|Z$  does not hold. Likewise, we denote by  $X \perp_p Y|Z$  (resp.  $X \not\perp_p Y|Z$ ) that  $X$  is independent (resp. dependent) of  $Y$  given  $Z$  in a probability distribution  $p$ . The independence model induced by  $G$ , denoted as  $I(G)$ , is the set of separation statements  $X \perp_G Y|Z$ . We say that  $p$  is Markovian with respect to  $G$  when  $X \perp_p Y|Z$  if  $X \perp_G Y|Z$  for all  $X$ ,  $Y$  and  $Z$  disjoint subsets of  $V$ . We say that  $p$  is faithful to  $G$  when  $X \perp_p Y|Z$  iff  $X \perp_G Y|Z$  for all  $X$ ,  $Y$  and  $Z$  disjoint subsets of  $V$ . If two CGs  $G$  and  $H$  are triplex equivalent, then

<sup>1</sup>See (Andersson et al., 2001, Remark 3.1) for the equivalence of this and the standard definition of separation.

$$I(G) = I(H).$$

Let  $X$ ,  $Y$ ,  $Z$  and  $W$  denote four disjoint subsets of  $V$ . Any probability distribution  $p$  satisfies the following properties: Symmetry  $X \perp_p Y|Z \Rightarrow Y \perp_p X|Z$ , decomposition  $X \perp_p Y \cup W|Z \Rightarrow X \perp_p Y|Z$ , weak union  $X \perp_p Y \cup W|Z \Rightarrow X \perp_p Y|Z \cup W$ , and contraction  $X \perp_p Y|Z \cup W \wedge X \perp_p W|Z \Rightarrow X \perp_p Y \cup W|Z$ . Moreover, if  $p$  is faithful to a CG, then it also satisfies the following properties: Intersection  $X \perp_p Y|Z \cup W \wedge X \perp_p W|Z \cup Y \Rightarrow X \perp_p Y \cup W|Z$ , and composition  $X \perp_p Y|Z \wedge X \perp_p W|Z \Rightarrow X \perp_p Y \cup W|Z$ .<sup>2</sup>

### 3 The Algorithm

Our algorithm, which can be seen in Table 1, resembles the well-known PC algorithm (Meek, 1995; Spirtes et al., 1993). It consists of two phases: The first phase (lines 1-8) aims at learning adjacencies, whereas the second phase (lines 9-10) aims at directing some of the adjacencies learnt. Specifically, the first phase declares that two nodes are adjacent iff they are not separated by any set of nodes. Note that the algorithm does not test every possible separator (see line 5). Note also that the separators tested are tested in increasing order of size (see lines 2, 5 and 8). The second phase consists of two steps. In the first step, the ends of some of the edges learnt in the first phase are blocked according to the rules R1-R4 in Table 2. A block is represented by a perpendicular line such as in  $\perp$  or  $\dashv$ , and it means that the edge cannot be directed in that direction. In the second step, the edges with exactly one unblocked end get directed in the direction of the unblocked end. The rules R1-R4 work as follows: If the conditions in the antecedent of a rule are satisfied, then the modifications in the consequent of the

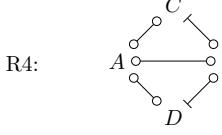
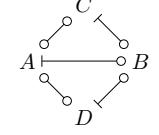
<sup>2</sup>To see it, note that there are Gaussian distributions  $p$  and  $q$  that are faithful to  $G$  and  $H$ , respectively (Levitz et al., 2001, Theorem 6.1). Moreover,  $p$  and  $q$  are Markovian wrt  $H$  and  $G$ , respectively, by Andersson et al. (2001, Theorem 5) and Levitz et al. (2001, Theorem 4.1).

<sup>3</sup>To see it, note that there is a Gaussian distribution that is faithful to  $G$  (Levitz et al., 2001, Theorem 6.1). Moreover, every Gaussian distribution satisfies the intersection and composition properties (Studený, 2005, Proposition 2.1 and Corollary 2.4).

Table 1: The algorithm.

|                                                                               |                                                                                                                              |
|-------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| Input: A probability distribution $p$ that is faithful to an unknown CG $G$ . |                                                                                                                              |
| Output: A CG $H$ that is triplex equivalent to $G$ .                          |                                                                                                                              |
| 1                                                                             | Let $H$ denote the complete undirected graph                                                                                 |
| 2                                                                             | Set $l = 0$                                                                                                                  |
| 3                                                                             | Repeat while $l \leq  V  - 2$                                                                                                |
| 4                                                                             | For each ordered pair of nodes $A$ and $B$ in $H$ st $A \in ad_H(B)$ and $ [ad_H(A) \cup ad_H(ad_H(A))] \setminus B  \geq l$ |
| 5                                                                             | If there is some $S \subseteq [ad_H(A) \cup ad_H(ad_H(A))] \setminus B$ st $ S  = l$ and $A \perp_p B   S$ then              |
| 6                                                                             | Set $S_{AB} = S_{BA} = S$                                                                                                    |
| 7                                                                             | Remove the edge $A - B$ from $H$                                                                                             |
| 8                                                                             | Set $l = l + 1$                                                                                                              |
| 9                                                                             | Apply the rules R1-R4 to $H$ while possible                                                                                  |
| 10                                                                            | Replace every edge $\leftarrow$ ( $\rightarrow$ ) in $H$ with $\rightarrow$ ( $\leftarrow$ )                                 |

Table 2: The rules R1-R4.

|     |                                                                                    |                       |                                                                                     |
|-----|------------------------------------------------------------------------------------|-----------------------|-------------------------------------------------------------------------------------|
| R1: | $A \circ \circ \circ B \circ \circ \circ C$<br>$\wedge B \notin S_{AC}$            | $\Rightarrow$         | $A \leftarrow \circ B \circ \circ \leftarrow C$                                     |
| R2: | $A \leftarrow \circ B \circ \circ \circ C$<br>$\wedge B \in S_{AC}$                | $\Rightarrow$         | $A \leftarrow \circ B \leftarrow \circ C$                                           |
| R3: | $A \circ \circ \circ \cdots \circ B$                                               | $\Rightarrow$         | $A \leftarrow \circ \cdots \circ B$                                                 |
| R4: |  | $\wedge A \in S_{CD}$ |  |

rule are applied. Note that the ends of some of the edges in the rules are labeled with a circle such as in  $\leftarrow\circ$  or  $\circ\circ$ . The circle represents an unspecified end, i.e. a block or nothing. The modifications in the consequents of the rules consist in adding some blocks. Note that only the blocks that appear in the consequents are added, i.e. the circled ends do not get modified. The conditions in the antecedents of R1, R2 and R4 consist of an induced subgraph of  $H$  and the fact that some of its nodes are or are not in some separators found in line 6. The condition in the antecedent of R3 is slightly different as it only says that there is a cycle in  $H$  whose edges have certain blocks, i.e. it says nothing about the subgraph induced by the nodes in the cycle or whether these nodes belong to some separators or not. Note that, when considering the application of R3, one does not need to consider intersecting cycles, i.e. cycles containing repeated nodes other than the initial and final ones.

#### 4 Correctness of the Algorithm

In this section, we prove that our algorithm is correct, i.e. it returns a CG the given probability distribution is faithful to. We start proving a result for any probability distribution that satisfies the intersection and composition properties.

Recall that any probability distribution that is faithful to a CG satisfies these properties and, thus, the following result applies to it.

**Lemma 1.** *Let  $p$  denote a probability distribution that satisfies the intersection and composition properties. Then,  $p$  is Markovian wrt a CG  $G$  iff  $p$  satisfies the following conditions:*

- C1:  $A \perp_p co_G(A) \setminus A \setminus ne_G(A) | pa_G(A \cup ne_G(A)) \cup ne_G(A)$  for all  $A \in V$ , and
- C2:  $A \perp_p V \setminus A \setminus deg(A) \setminus pa_G(A) | pa_G(A)$  for all  $A \in V$ .

*Proof.* It follows from Andersson et al. (2001, Theorem 3) and Levitz et al. (2001, Theorem 4.1) that  $p$  is Markovian wrt  $G$  iff  $p$  satisfies the following conditions:

- L1:  $A \perp_p co_G(A) \setminus A \setminus ne_G(A) | [V \setminus co_G(A) \setminus deg(co_G(A))] \cup ne_G(A)$  for all  $A \in V$ , and
- L2:  $A \perp_p V \setminus co_G(A) \setminus deg(co_G(A)) \setminus pa_G(A) | pa_G(A)$  for all  $A \in V$ .

Clearly, C2 holds iff L2 holds because  $deg(A) = [co_G(A) \cup deg(co_G(A))] \setminus A$ . We prove below that if L2 holds, then C1 holds iff L1 holds. We first prove the if part.

1.  $B \perp_p V \setminus co_G(B) \setminus de_G(co_G(B)) \setminus pa_G(B)|pa_G(B)$  for all  $B \in A \cup ne_G(A)$  by L2.
2.  $B \perp_p V \setminus co_G(B) \setminus de_G(co_G(B)) \setminus pa_G(A \cup ne_G(A))|pa_G(A \cup ne_G(A))$  for all  $B \in A \cup ne_G(A)$  by weak union on 1.
3.  $A \cup ne_G(A) \perp_p V \setminus co_G(A) \setminus de_G(co_G(A)) \setminus pa_G(A \cup ne_G(A))|pa_G(A \cup ne_G(A))$  by repeated application of symmetry and composition on 2.
4.  $A \perp_p V \setminus co_G(A) \setminus de_G(co_G(A)) \setminus pa_G(A \cup ne_G(A))|pa_G(A \cup ne_G(A)) \cup ne_G(A)$  by symmetry and weak union on 3.
5.  $A \perp_p co_G(A) \setminus A \setminus ne_G(A)|[V \setminus co_G(A) \setminus de_G(co_G(A))] \cup ne_G(A)$  by L1.
6.  $A \perp_p [co_G(A) \setminus A \setminus ne_G(A)] \cup [V \setminus co_G(A) \setminus de_G(co_G(A)) \setminus pa_G(A \cup ne_G(A))]|pa_G(A \cup ne_G(A)) \cup ne_G(A)$  by contraction on 4 and 5.
7.  $A \perp_p co_G(A) \setminus A \setminus ne_G(A)|pa_G(A \cup ne_G(A)) \cup ne_G(A)$  by decomposition on 6.

We now prove the only if part.

8.  $A \perp_p co_G(A) \setminus A \setminus ne_G(A)|pa_G(A \cup ne_G(A)) \cup ne_G(A)$  by C1.
9.  $A \perp_p [V \setminus co_G(A) \setminus de_G(co_G(A)) \setminus pa_G(A \cup ne_G(A))] \cup [co_G(A) \setminus A \setminus ne_G(A)]|pa_G(A \cup ne_G(A)) \cup ne_G(A)$  by composition on 4 and 8.
10.  $A \perp_p co_G(A) \setminus A \setminus ne_G(A)|[V \setminus co_G(A) \setminus de_G(co_G(A))] \cup ne_G(A)$  by weak union on 9.

□

**Lemma 2.** After line 8,  $G$  and  $H$  have the same adjacencies.

*Proof.* Consider any pair of nodes  $A$  and  $B$  in  $G$ . If  $A \in ad_G(B)$ , then  $A \not\perp_p B|S$  for all  $S \subseteq V \setminus [A \cup B]$  by the faithfulness assumption. Consequently,  $A \in ad_H(B)$  at all times. On the other hand, if  $A \notin ad_G(B)$ , then consider the following cases.

**Case 1** Assume that  $co_G(A) = co_G(B)$ . Then,  $A \perp_p co_G(A) \setminus A \setminus ne_G(A)|pa_G(A \cup ne_G(A)) \cup ne_G(A)$  by C1 in Lemma 1 and, thus,

$A \perp_p B|pa_G(A \cup ne_G(A)) \cup ne_G(A)$  by decomposition and  $B \notin ne_G(A)$ , which follows from  $A \notin ad_G(B)$ . Note that, as shown above,  $pa_G(A \cup ne_G(A)) \cup ne_G(A) \subseteq [ad_H(A) \cup ad_H(ad_H(A))] \setminus B$  at all times.

**Case 2** Assume that  $co_G(A) \neq co_G(B)$ . Then,  $A \notin de_G(B)$  or  $B \notin de_G(A)$  because  $G$  has no semidirected cycle. Assume without loss of generality that  $B \notin de_G(A)$ . Then,  $A \perp_p V \setminus A \setminus de_G(A) \setminus pa_G(A)|pa_G(A)$  by C2 in Lemma 1 and, thus,  $A \perp_p B|pa_G(A)$  by decomposition,  $B \notin de_G(A)$ , and  $B \notin pa_G(A)$  which follows from  $A \notin ad_G(B)$ . Note that, as shown above,  $pa_G(A) \subseteq ad_H(A) \setminus B$  at all times.

Therefore, in either case, there will exist some  $S$  in line 5 such that  $A \perp_p B|S$  and, thus, the edge  $A - B$  will be removed from  $H$  in line 7. Consequently,  $A \notin ad_H(B)$  after line 8. □

The next lemma proves that the rules R1-R4 are sound, i.e. if the antecedent holds in  $G$ , then so does the consequent.

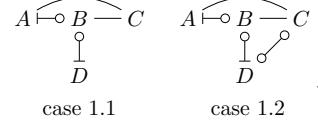
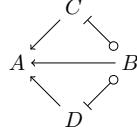
**Lemma 3.** The rules R1-R4 are sound.

*Proof.* According to the antecedent of R1,  $G$  has a triplex  $(\{A, C\}, B)$ . Then,  $G$  has an induced subgraph of the form  $A \rightarrow B \leftarrow C$ ,  $A \rightarrow B - C$  or  $A - B \leftarrow C$ . In either case, the consequent of R1 holds.

According to the antecedent of R2, (i)  $G$  does not have a triplex  $(\{A, C\}, B)$ , (ii)  $A \rightarrow B$  or  $A - B$  is in  $G$ , (iii)  $B \in ad_G(C)$ , and (iv)  $A \notin ad_G(C)$ . Then,  $B \rightarrow C$  or  $B - C$  is in  $G$ . In either case, the consequent of R2 holds.

According to the antecedent of R3, (i)  $G$  has a descending route from  $A$  to  $B$ , and (ii)  $A \in ad_G(B)$ . Then,  $A \rightarrow B$  or  $A - B$  is in  $G$ , because  $G$  has no semidirected cycle. In either case, the consequent of R3 holds.

According to the antecedent of R4, neither  $B \rightarrow C$  nor  $B \rightarrow D$  are in  $G$ . Assume to the contrary that  $A \leftarrow B$  is in  $G$ . Then,  $G$  must have an induced subgraph that is consistent with



because, otherwise, it would have a semidirected cycle. However, this induced subgraph contradicts that  $A \in S_{CD}$ .  $\square$

**Lemma 4.** *After line 10, G and H have the same triplexes. Moreover, H has all the immoralities in G.*

*Proof.* We first prove that any triplex in  $H$  is in  $G$ . Assume to the contrary that  $H$  has a triplex  $(\{A, C\}, B)$  that is not in  $G$ . This is possible iff, when line 10 is executed,  $H$  has an induced subgraph of one of the following forms:

$$A \rightarrow B \leftarrow C \quad A \leftarrow B \rightarrow C \quad A \leftarrow B \leftarrow C$$

where  $B \in S_{AC}$  by Lemma 2. The first and second forms are impossible because, otherwise,  $A \rightsquigarrow B$  would be in  $H$  by R2. The third form is impossible because, otherwise,  $B \leftarrow C$  would be in  $H$  by R2.

We now prove that any triplex  $(\{A, C\}, B)$  in  $G$  is in  $H$ . Let the triplex be of the form  $A \rightarrow B \leftarrow C$ . Then, when line 10 is executed,  $A \rightsquigarrow B \rightsquigarrow C$  is in  $H$  by R1, and neither  $A \leftarrow B$  nor  $B \leftarrow C$  is in  $H$  by Lemmas 2 and 3. Then, the triplex is in  $H$ . Note that the triplex is an immorality in both  $G$  and  $H$ . Likewise, let the triplex be of the form  $A \rightarrow B - C$ . Then, when line 10 is executed,  $A \rightsquigarrow B \rightsquigarrow C$  is in  $H$  by R1, and  $A \leftarrow B$  is not in  $H$  by Lemmas 2 and 3. Then, the triplex is in  $H$ . Note that the triplex is a flag in  $G$  but it may be an immorality in  $H$ .  $\square$

**Lemma 5.** *After line 9, H does not have any induced subgraph of the form  $A \rightsquigarrow B \rightsquigarrow C$ .*

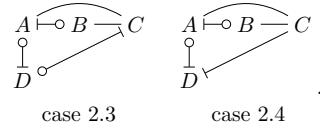
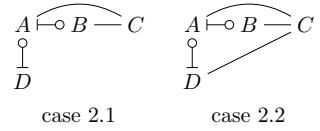
*Proof.* Assume to the contrary that the lemma does not hold. Consider the following cases.

**Case 1** Assume that  $A \rightsquigarrow B$  is in  $H$  due to R1. Then, when R1 was applied,  $H$  had an induced subgraph of one of the following forms:

**Case 1.1** If  $B \notin S_{CD}$  then  $B \rightarrow C$  is in  $H$  by R1, else  $B \leftarrow C$  is in  $H$  by R2. Either case is a contradiction.

**Case 1.2** If  $C \notin S_{AD}$  then  $A \leftarrow C$  is in  $H$  by R1, else  $B \rightarrow C$  is in  $H$  by R4. Either case is a contradiction.

**Case 2** Assume that  $A \rightsquigarrow B$  is in  $H$  due to R2. Then, when R2 was applied,  $H$  had an induced subgraph of one of the following forms:



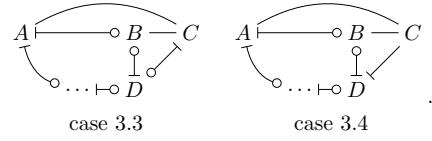
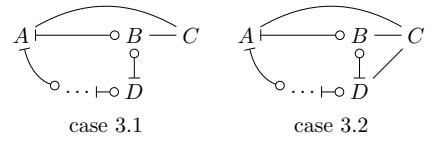
**Case 2.1** If  $A \notin S_{CD}$  then  $A \rightarrow C$  is in  $H$  by R1, else  $A \leftarrow C$  is in  $H$  by R2. Either case is a contradiction.

**Case 2.2** Restart the proof with  $D$  instead of  $A$  and  $A$  instead of  $B$ .

**Case 2.3** Then,  $A \rightarrow C$  is in  $H$  by R3, which is a contradiction.

**Case 2.4** If  $C \notin S_{BD}$  then  $B \leftarrow C$  is in  $H$  by R1, else  $B \rightarrow C$  is in  $H$  by R2. Either case is a contradiction.

**Case 3** Assume that  $A \rightsquigarrow B$  is in  $H$  due to R3. Then, when R3 was applied,  $H$  had an induced subgraph of one of the following forms:



Note that  $C$  cannot belong to the route  $A \rightarrow \dots \rightarrow D$  because, otherwise,  $A \rightarrow C$  would be in  $H$  by R3.

**Case 3.1** If  $B \notin S_{CD}$  then  $B \rightarrow C$  is in  $H$  by R1, else  $B \leftarrow C$  is in  $H$  by R2.

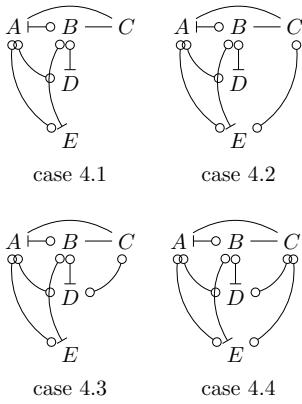
Either case is a contradiction.

**Case 3.2** Restart the proof with  $D$  instead of  $A$ .

**Case 3.3** Then,  $B \rightarrow C$  is in  $H$  by R3, which is a contradiction.

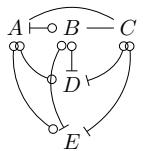
**Case 3.4** Then,  $A \leftarrow C$  is in  $H$  by R3, which is a contradiction.

**Case 4** Assume that  $A \rightarrow B$  is in  $H$  due to R4. Then, when R4 was applied,  $H$  had an induced subgraph of one of the following forms:



**Cases 4.1-4.3** If  $B \notin S_{CD}$  or  $B \notin S_{CE}$  then  $B \rightarrow C$  is in  $H$  by R1, else  $B \leftarrow C$  is in  $H$  by R2. Either case is a contradiction.

**Case 4.4** Assume that  $C \in S_{DE}$ . Then,  $B \rightarrow C$  is in  $H$  by R4, which is a contradiction. On the other hand, assume that  $C \notin S_{DE}$ . Then, it follows from applying R1 that  $H$  has an induced subgraph of the form



Note that  $A \in S_{DE}$  because, otherwise, R4 would not have been applied.

Then,  $A \leftarrow C$  is in  $H$  by R4, which is a contradiction.  $\square$

**Lemma 6.** After line 9, every cycle in  $H$  that has an edge  $\leftarrow$  also has an edge  $\rightarrow$ .

*Proof.* Assume to the contrary that  $H$  has a cycle  $\rho : V_1, \dots, V_n = V_1$  that has an edge  $\leftarrow$  but no edge  $\rightarrow$ . Note that every edge in  $\rho$  cannot be  $\rightarrow$  because, otherwise, every edge in  $\rho$  would be  $\rightarrow$  by repeated application of R3, which contradicts the assumption that  $\rho$  has an edge  $\leftarrow$ . Therefore,  $\rho$  has an edge  $-$  or  $\rightarrow$ . Since the latter contradicts the assumption that the lemma does not hold,  $\rho$  has an edge  $-$ . Assume that  $\rho$  is of length three. Then,  $\rho$  is of one of the following forms:

$$V_1 \leftarrow V_2 \rightarrow V_3 \quad V_1 \leftarrow V_2 \rightarrow V_3 \quad V_1 \leftarrow V_2 \leftarrow V_3 .$$

The first form is impossible by Lemma 5. The second form is impossible because, otherwise,  $V_2 \rightarrow V_3$  would be in  $H$  by R3. The third form is impossible because, otherwise,  $V_1 \leftarrow V_3$  would be in  $H$  by R3. Thus, the lemma holds for cycles of length three.

Assume that  $\rho$  is of length greater than three. Recall from above that  $\rho$  has an edge  $-$  and no edge  $\rightarrow$ . Let  $V_{i+1} - V_{i+2}$  be the first edge  $-$  in  $\rho$ . Assume without loss of generality that  $i > 0$ . Then,  $\rho$  has a subpath of the form  $V_i \rightarrow V_{i+1} - V_{i+2}$ . Note that  $V_i \in ad_H(V_{i+2})$  because, otherwise, if  $V_{i+1} \notin S_{V_i V_{i+2}}$  then  $V_{i+1} \rightarrow V_{i+2}$  would be in  $H$  by R1, else  $V_{i+1} \leftarrow V_{i+2}$  would be in  $H$  by R2. Thus,  $H$  has an induced subgraph of one of the following forms:

$$V_i \leftarrow V_{i+1} - V_{i+2} \quad V_i \leftarrow V_{i+1} - V_{i+2} \quad V_i \leftarrow V_{i+1} - V_{i+2} .$$

The first form is impossible by Lemma 5. The second form is impossible because, otherwise,  $V_{i+1} \rightarrow V_{i+2}$  would be in  $H$  by R3. Thus, the third form is the only possible. Note that this implies that  $\varrho : V_1, \dots, V_i, V_{i+2}, \dots, V_n = V_1$  is a cycle in  $H$  that has an edge  $\leftarrow$  and no edge  $\rightarrow$ .

By repeatedly applying the reasoning above, one can see that  $H$  has a cycle of length three

that has an edge  $\leftarrow$  and no edge  $\rightarrow$ . As shown above, this is impossible. Thus, the lemma holds for cycles of length greater than three too.  $\square$

**Theorem 1.** *After line 10,  $H$  is triplex equivalent to  $G$  and it has no semidirected cycle.*

*Proof.* Lemma 2 implies that  $G$  and  $H$  have the same adjacencies. Lemma 4 implies that  $G$  and  $H$  have the same triplexes. Lemma 6 implies that  $H$  has no semidirected cycle.  $\square$

## 5 Discussion

In this paper, we have presented an algorithm for learning an AMP CG a given probability distribution  $p$  is faithful to. In practice, of course, we do not usually have access to  $p$  but to a finite sample from it. Our algorithm can easily be modified to deal with this situation: Replace  $A \perp_p B | S$  in line 5 with a hypothesis test, preferably with one that is consistent so that the resulting algorithm is asymptotically correct.

It is worth mentioning that, whereas R1, R2 and R4 only involve three or four nodes, R3 may involve many more. Hence, it would be desirable to replace R3 with a simpler rule such as

$$A \overset{\text{?}}{\circ} B \overset{\text{?}}{\circ} C \Rightarrow A \overset{\text{?}}{\circ} B \overset{\text{?}}{\circ} C.$$

Unfortunately, we have not succeeded so far in proving the correctness of our algorithm with such a simpler rule. Note that the output of our algorithm will be the same whether we keep R3 or we replace it with a simpler sound rule. The only benefit of the simpler rule may be a decrease in running time.

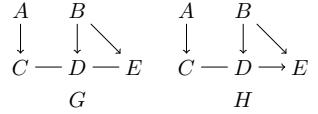
We have shown in Lemma 4 that, after line 10,  $H$  has all the immoralities in  $G$  or, in other words, every flag in  $H$  is in  $G$ . The following lemma strengthens this fact.

**Lemma 7.** *After line 10, every flag in  $H$  is in every CG  $F$  that is triplex equivalent to  $G$ .*

*Proof.* Note that every flag in  $H$  is due to an induced subgraph of the form  $A \leftarrow B \leftarrow C$ . Note also that all the blocks in  $H$  follow from the

adjacencies and triplexes in  $G$  by repeated application of R1-R4. Since  $G$  and  $F$  have the same adjacencies and triplexes, all the blocks in  $H$  hold in both  $G$  and  $F$  by Lemma 3.  $\square$

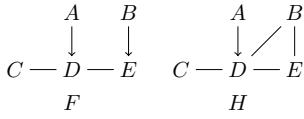
The lemma above implies that, in terms of Roverato and Studený (2006), our algorithm outputs a deflagged graph. Roverato and Studený (2006) also introduce the concept of strongly equivalent CGs: Two CGs are strongly equivalent iff they have the same adjacencies, immoralities and flags. Unfortunately, not every edge  $\rightarrow$  in  $H$  after line 10 is in every deflagged graph that is triplex equivalent to  $G$ , as the following example illustrates, where both  $G$  and  $H$  are deflagged graphs.



Therefore, in terms of Roverato and Studený (2006), our algorithm outputs a deflagged graph but not the largest deflagged graph. The latter is a distinguished member of a class of triplex equivalent CGs. Fortunately, the largest deflagged graph can easily be obtained from any deflagged graph in the class (Roverato and Studený, 2006, Corollary 17).

The correctness of our algorithm lies upon the assumption that  $p$  is faithful to some CG. This is a strong requirement that we would like to weaken, e.g. by replacing it with the milder assumption that  $p$  satisfies the composition property. Correct algorithms for learning directed and acyclic graphs (a.k.a Bayesian networks) under the composition property assumption exist (Chickering and Meek, 2002; Nielsen et al., 2003). We have recently developed a correct algorithm for learning LWF CGs under the composition property (Peña et al., 2012). The way in which these algorithms proceed (a.k.a. score+search based approach) is rather different from that of the algorithm presented in this paper (a.k.a. constraint based approach). In a nutshell, they can be seen as consisting of two phases: A first phase that starts from the empty graph  $H$  and adds single edges to

it until  $p$  is Markovian wrt  $H$ , and a second phase that removes single edges from  $H$  until  $p$  is Markovian wrt  $H$  and  $p$  is not Markovian wrt any CG  $F$  st  $I(H) \subseteq I(F)$ . The success of the first phase is guaranteed by the composition property assumption, whereas the success of the second phase is guaranteed by the so-called Meek's conjecture (Meek, 1997). Specifically, given two directed and acyclic graphs  $F$  and  $H$  st  $I(H) \subseteq I(F)$ , Meek's conjecture states that we can transform  $F$  into  $H$  by a sequence of operations st, after each operation,  $F$  is a directed and acyclic graph and  $I(H) \subseteq I(F)$ . The operations consist in adding a single edge to  $F$ , or replacing  $F$  with a triplex equivalent directed and acyclic graph. Meek's conjecture was proven to be true in (Chickering, 2002, Theorem 4). The extension of Meek's conjecture to LWF CGs was proven to be true in (Peña, 2011, Theorem 1). Unfortunately, the extension of Meek's conjecture to AMP CGs does not hold, as the following example illustrates.



Then,  $I(H) = \{X \perp_H Y | Z : X \perp_H Y | Z \in I_1(H) \cup I_2(H) \vee Y \perp_H X | Z \in I_1(H) \cup I_2(H)\}$  where  $I_1(H) = \{A \perp_H Y | Z : Y, Z \subseteq B \cup C \cup E \wedge D \notin Z\}$  and  $I_2(H) = \{C \perp_H Y | Z : Y \subseteq B \cup E \wedge A \cup D \subseteq Z\}$ . One can easily confirm that  $I(H) \subseteq I(F)$  by using the definition of separation. However, there is no CG that is triplex equivalent to  $F$  or  $H$  and, obviously, one cannot transform  $F$  into  $H$  by adding a single edge.

While the example above compromises the development of score+search learning algorithms that are correct and efficient under the composition property assumption, it is not clear to us whether it also does it for constraint based algorithms. This is something we plan to study.

## Acknowledgments

We would like to thank the anonymous Reviewers and specially Reviewer 2 for their comments. This work is funded by the Center for Indus-

trial Information Technology (CENIIT) and a so-called career contract at Linköping University, and by the Swedish Research Council (ref. 2010-4808).

## References

- Andersson, S. A., Madigan, D. and Perlman, M. D. Alternative Markov Properties for Chain Graphs. *Scandinavian Journal of Statistics*, 28:33-85, 2001.
- Chickering, D. M. Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Research*, 3:507-554, 2002.
- Chickering, D. M. and Meek, C. Finding Optimal Bayesian Networks. In *Proceedings of 18th Conference on Uncertainty in Artificial Intelligence*, 94-102, 2002.
- Drton, M. and Eichler, M. Maximum Likelihood Estimation in Gaussian Chain Graph Models under the Alternative Markov Property. *Scandinavian Journal of Statistics*, 33:247-257, 2006.
- Lauritzen, S. L. *Graphical Models*. Oxford University Press, 1996.
- Levitz, M., Perlman M. D. and Madigan, D. Separation and Completeness Properties for AMP Chain Graph Markov Models. *The Annals of Statistics*, 29:1751-1784, 2001.
- Ma, Z., Xie, X. and Geng, Z. Structural Learning of Chain Graphs via Decomposition. *Journal of Machine Learning Research*, 9:2847-2880, 2008.
- Meek, C. Causal Inference and Causal Explanation with Background Knowledge. *Proceedings of 11th Conference on Uncertainty in Artificial Intelligence*, 403-418, 1995.
- Meek, C. *Graphical Models: Selecting Causal and Statistical Models*. PhD thesis, Carnegie Mellon University, 1997.
- Nielsen, J. D., Kočka, T. and Peña, J. M. On Local Optima in Learning Bayesian Networks. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 435-442, 2003.
- Peña, J. M. Towards Optimal Learning of Chain Graphs. arXiv:1109.5404v1 [stat.ML], 2011.
- Peña, J. M., Sonntag, D. and Nielsen, J. D. An Inclusion Optimal Algorithm for Chain Graph Structure Learning. Submitted, 2012.
- Roverato, A. and Studený, M. A Graphical Representation of Equivalence Classes of AMP Chain Graphs. *Journal of Machine Learning Research*, 7:1045-1078, 2006.
- Spirites, P., Glymour, C. and Scheines, R. *Causation, Prediction, and Search*. Springer-Verlag, 1993.
- Studený, M. A Recovery Algorithm for Chain Graphs. *International Journal of Approximate Reasoning*, 17:265-293, 1997.
- Studený, M. *Probabilistic Conditional Independence Structures*. Springer, 2005.

# Prediction of Coffee Rust Disease Using Bayesian Networks

Cora B. Pérez-Ariza

Universidad de Granada, Spain (cora@decsai.ugr.es)

Ann E. Nicholson

Monash University, Australia (ann.nicholson@monash.edu)

M. Julia Flores

University of Castilla-La Mancha (julia.flores@uclm.es)

## Abstract

In this paper we present an agricultural case study for learning Bayesian networks (BNs), namely prediction of coffee rust. Wide-spread in all major production areas, coffee rust causes premature defoliation, weakening the plant and reducing subsequent yield. It is typically controlled by use of chemical fungicides which must be applied before symptoms of infection are observed. Improved prediction would reduce the use of fungicides, producing healthier quality product and decreasing both economic costs and environmental impact.

We use a dataset obtained from an experimental farm in Brazil over 8 years. Our preliminary data analysis informed our pre-processing of the original dataset. We also identified a number of structural priors, which our BN learner, CaMML (Causal Minimum Message Length), incorporates into its scoring metric and hence into its structure learning. Previous research has applied other classification methods to this coffee rust dataset. We compare the results from a range of BNs learnt by CaMML with these previous approaches. The incorporation of structural priors in the BN learning yielded better models in terms of accuracy and interpretability. Although the BN's predictions were comparable with some of the other techniques, they were clearly worse than decision trees, which seem to be taking advantage of context sensitive cases; this suggests avenues for improving the quality of the BN models.

## 1 Introduction

In this work we present an application of Bayesian networks to an agricultural case study, namely the prediction of coffee rust. This is a devastating disease for coffee plantations, as it damages plants and reduces their yield through premature fall of infected leaves. Coffee rust has provoked intensive losses in many coffee-producing countries. It attacks all coffee species, being more severe in *Coffea arabica*. The disease first appeared in 1970 in the occidental hemisphere, in Brazil, and was rapidly extended to other major coffee producing countries in Central and South America, whose coffee trees are more sensitive to rust in-

fection. Coffee rust is primarily caused by a fungus called *Hemileia vastatrix*. The symptoms of coffee rust infection include the appearance of spots on the upper leaf surface of plants. Its incidence can be measured by the percentage of leaves infected by the fungus.

In Brazil, damage from coffee rust leads to yield reduction of up to 35% in regions where climate conditions are propitious to the disease. The impact is thus considerable due to the economic importance of the coffee crop. The traditional way to prevent the disease is to apply agrochemical fungicides on fixed calendar dates. However, the fungicides both contaminate the environment and reduce the quality of the cof-

fee. Moreover, as the intensity of the disease between seasons is subject to major variations, the use of agrochemicals is not always justified. The prevalence and impact of coffee rust have lead to many studies about it, including its prediction and how to decide whether to apply fungicides. Plant pathologists have tried to characterise this infection forecasting (Kushalappa, 1990), and more recently machine learning researchers in Brazil have applied decision trees (Meira et al., 2008), regression Support Vector Machines (SVM) (Luaces et al., 2010) and non-deterministic classifiers (Luaces et al., 2011) (described in more detail in Section 2), to the problem; but it has not yet been solved.

Bayesian networks have been applied to many real applications in environmental sciences, in fields such as farming, water resources, reforestation and ecological modelling; agricultural examples include fungicide for mildew in wheat (Jensen, 1995) and growing barley without pesticides (Kristensen and Rasmussen, 2002). Aguilera et al. (2011) have recently reviewed BN applications in the environmental sciences published between 1990 and 2010. This background encouraged us to apply Bayesian networks to this particular agricultural case study, the prediction of coffee rust.

BNs may be built either by eliciting expert knowledge or by automated causal discovery. With the availability of a dataset, used by the aforementioned Brazilian machine learning (ML) researchers, our approach here is focused on ML. The dataset comprises monthly accounts of the incidence of disease on an experimental farm in Brazil over 8 years. Additionally, the dataset registers the values of variables known to stimulate the growth of fungus (Avelino et al., 2006): weather conditions, fruit load of the plantation, and spacing between plants. We describe the dataset and the pre-processing undertaken (e.g. variable selection, discretization) in Section 4.

In this paper, we perform experiments with a set of BNs (Section 5). We first construct two simple handcrafted BNs (ignoring the weather variables) for use as baseline models. Then we learn the BN from data, using the CaMML

(Causal Minimum Message Length) BN learner (Section 3), on both the full dataset and a reduced dataset after applying variable selection. We also run CaMML with different structural priors, which constrain its search across possible models. In Section 6, we compare the hand-crafted and learned BNs against standard prediction methods available in Weka (Witten and Frank, 2011) (Naive Bayes and decision trees), and against the reported results from the Brazilian researchers.

## 2 Previous ML approaches

We now briefly describe the ML approaches previously applied to the Brazilian coffee rust dataset, as this informs the modelling choices we have taken with dataset pre-processing (described in Section 4), and provides an additional comparison for the results given in Section 5.

In (Meira et al., 2008), the authors develop a decision tree with the aim of aiding the understanding of coffee rust epidemics. The class variable was the change in infection rate, defined as the percentage of infected leafs, based on the monthly observations. They used three classes to classify the monthly change in the infection: reduction or stagnation ( $\leq 0\%$ ); moderate growth ( $> 0 \& \leq 5\%$ ); and accelerated growth ( $> 5\%$ ). They used 14 predictive variables: space, load, and weather measurements related to temperature, rain and relative humidity. The model correctly classified 73% using cross-validation, while the success rates were 88%, 57% and 79%, respectively, for these infection rate classes (i.e. the model did poorly on the moderate increase in incidence). The most important explanatory variables were found to be mean temperature during leaf wetness periods, expected yield, mean of maximum temperatures during the incubation period and relative air humidity.

More recently, these researchers presented a more sophisticated approach to the same problem (and dataset), using SVM regression (Luaces et al., 2010); rather than predicting the change in incidence, they consider actual incidence. They trained a SVM that provided a correlation of about 0.94 between predicted and ac-

tual incidences. However, when they try to devise an alarm system for predicting values above a given threshold, the number of false negatives turned to be too high (i.e. they were missing too many cases). So, they implemented what they call *nondeterministic* regressors, changing target points for intervals of fixed width. This allowed them to consider three possibilities: alarm, non-alarm and warnings (within a variable window around 4.5% incidence). This lowered the number of false negatives but, naturally, added false warnings.

Finally, (Luaces et al., 2011) is a further extension of this work to include additional nondeterministic classifiers (regressor SVM was adapted interpreting intervals as classes) defined by means of an optimization problem. To perform comparison between these alarm predictors, they present a framework where the costs of false negatives are higher than that of false positives, and both are higher than the cost of warning predictions. Under mild conditions they identified parameter settings, i.e. ranges of costs, where one of the nondeterministic classifiers outperforms the other learners.

In this paper we will analyse the behaviour of Bayesian networks for the coffee rust prediction problem. The advantages of BNs are (1) they are able to capture (in)dependencies existing among the variables involved, and the use of conditional probabilities make them capable of inherently dealing with uncertainty; (2) the graphical representation of relationships between variables facilitates the interpretation and formulation of conclusions about the domain of study; and (3) BNs can combine causal relationships with probabilistic logic, which helps to incorporate expert knowledge into the model. Next we describe the capabilities that the CaMML BN learner offers to do so.

### 3 The CaMML BN Learner

CaMML attempts to learn the best causal structure to account for the data, using a minimum message length (MML) metric (Wallace, 2005) with a two-phase search, simulated annealing followed by Markov Chain Monte Carlo search, over the model space. CaMML has

been developed at Monash University over the past 16 years; see (Korb and Nicholson, 2010, Ch 9) for a full description. We used the version of most recent open-source version of CaMML,<sup>1</sup> which requires variables to be discretized. CaMML supports multiple ways of describing prior information about relationships between variables (O'Donnell et al., 2006), each of which can be accompanied by a confidence level. The types of priors are: full structure, direct causal connections, direct relation of unknown direction, causal dependency (ancestor relationship), correlation and temporal order (tiers) ( $\{A_1, A_2, \dots, A_m\} \prec \{B_1, B_2, \dots, B_n\}$ ). In tiers, a partial ordering constrains arcs, based on the notion that tiers separate the variables on a timeline and that causality only occurs forward in time; this is standard in BN learning (e.g. Tetrads IV and K2). Thus  $\prec$  means that the variables  $A_i$  occur before the  $B_j$ , which for CaMML becomes the structural constraint the  $A_i$  cannot be descendants of the variables  $B_j$ .

The expert may provide CaMML with a full structure or any combination of the prior types above, together with their confidence in each prior (expressed as a probability).<sup>2</sup> CaMML then combines these using the MML encoding of each kind of structural prior and the confidence probabilities, together with the default arc probability. For the experiments reported here, we provide CaMML with temporal tier priors, described in Sec. 5.2 below.

## 4 Description of the dataset

The data used for this work was obtained from an experimental farm<sup>3</sup> (Fundação Procafé, Varginha, Minas Gerais, Brazil) where the incidence of coffee rust was monitored from October 1998 to October 2006. The farm is organized into eight plots, where the rust is observed in different combinations of spacing between plants (dense or thin) and different fruit

<sup>1</sup><https://github.com/rodneyodonnell/CaMML>

<sup>2</sup>Other BN learners also support the use of structural knowledge, but they have been limited to specifying one or two of these kinds of priors.

<sup>3</sup>This dataset, not yet publicly available, was provided to us by Luiz Henrique Rodrigues.

load (high or low). A meteorological station reports the temperature, humidity, wind speed, solar radiation and amount of rain registered every 30 minutes. Some metheorological information was missing in the original database due to failures in the station; we removed entries that contained missing data (as did (Meira et al., 2008), (Luaces et al., 2010) and (Luaces et al., 2011)). The rust observations were taken the first day of every month by picking 100 leaves from each plant of each plot and computing the average number of infected leaves.

We used the following data: (1) monthly information of the incidence of the coffee rust, and (2) a daily summary of all the weather observations taken that day.

Fig.1 shows the timeline for monthly observations, the possible timing of prediction (10, 15, 20, 25 or 30 days before next first day of the month), and the summary of daily weather conditions we used (the 45 days prior to the prediction day, as this is the potential period of infection). Note that for each target day, we know the incidence of coffee rust detected on the previous target day ( $I_{d0}$ ). After preprocessing, the final database comprises 1716 records of 34 variables. We ignore the time series aspect of the data and treat each record as independent and identically distributed (although we incorporate aspects of time through the month and year variables).

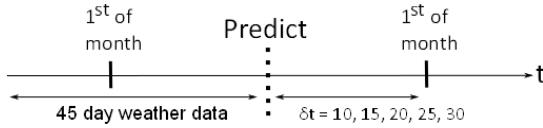


Figure 1: Sequence of prediction

First, we observed the incidence of coffee rust for all the registered months. Fig. 2 shows two examples of the time series (for high load plots, both thin and dense spacing); the periodicity of the coffee rust incidence is clear. The incidence of the rust over successive months (plotted in Fig. 3) is highly correlated (correlation coefficient  $\rho_{I_{d0}, I_{dt}} = 0.557157$ ). We use a threshold of 4.5% incidence to confirm an infection, the threshold typically used by farmers when deciding to chemically treat the plants, and used

in (Luaces et al., 2010; Luaces et al., 2011). However, we discretized (by hand) the incidence variables into 5 states: [0], {0-4}, {4-4.5}, {4.5-5} and {5-100}, in order to reflect different levels of infections (negative, warning and positive), around that threshold.

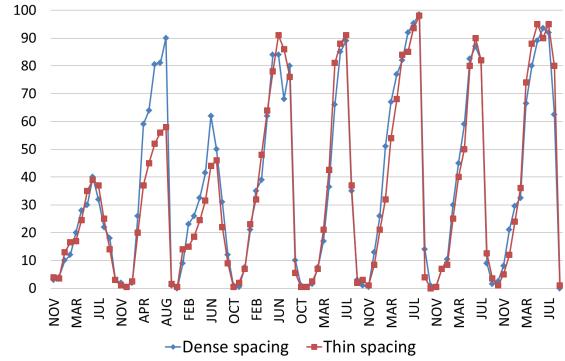


Figure 2: Monthly incidence of coffee rust for each month, in plots with highly loaded plants, for different spacings between plants

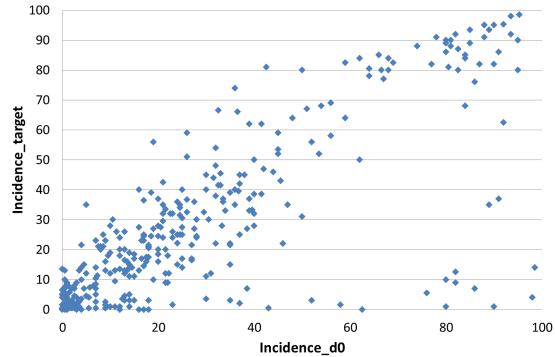


Figure 3: Correlation between the incidence of coffee rust and its previous presence

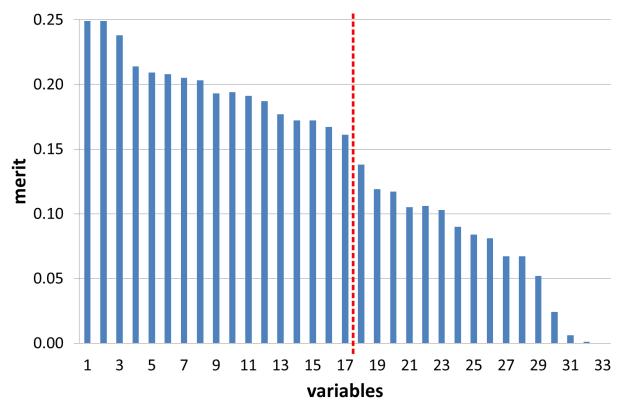


Figure 4: SU merit for the variables ranked.

Table 1: Variables used in the study. SU shows the attribution selection ranking, while  $\star$  indicates the variable was used in experiments with dataset $^r$ . The first block of variables corresponds to the incidence records, discretized by hand. The second block are all discrete variables, and the third block corresponds to the weather variables, all continuous and discretized using Weka.

| Variable             | Description                                                                          | SU | ( $\star$ ) |
|----------------------|--------------------------------------------------------------------------------------|----|-------------|
| Incidence_target     | Percentage of leaves infected on target day (%) class variable                       |    |             |
| Incidence_d0         | Percentage of leaves infected on d0 (%)                                              | 4  | $\star$     |
| Year                 | year of measure                                                                      | 29 |             |
| Month                | month of incidence                                                                   | 3  | $\star$     |
| Wet_season           | YES if incubation period starts in Feb, March or May. NO otherwise                   | 18 |             |
| Season               | Summer(Dec-Feb), Autumn(Mar-May), Winter June-Aug), Spring(Sept-Nov)                 | 1  | $\star$     |
| Load                 | fruit load of plants                                                                 | 30 |             |
| Spacing              | spacing between plants                                                               | 31 |             |
| Days_d0_now          | days between previous 1st of month and prediction date                               | 33 |             |
| Days_now_target      | 30, 25, 20, 15 & 10 days between prediction and next 1st of month                    | 32 |             |
| Tmed_avg             | average of temperatures during the period of infection (PINF) ( $^{\circ}C$ )        | 11 | $\star$     |
| Tmed_max             | maximum of temperature during the PINF ( $^{\circ}C$ )                               | 16 | $\star$     |
| Tmax_avg             | average of maximum temperatures during the PINF ( $^{\circ}C$ )                      | 25 |             |
| Tmax_max             | maximum of maximum temperatures during the PINF ( $^{\circ}C$ )                      | 20 |             |
| Tmin_avg             | average of minimum temperatures furing the PINF ( $^{\circ}C$ )                      | 12 | $\star$     |
| Tmin_min             | Minimum of minimum temperatures furing the PINF ( $^{\circ}C$ )                      | 10 | $\star$     |
| SolarRadiation_avg   | Average of accumulated solar radiation during PINF ( $W/m^2?$ )                      | 28 |             |
| SolarRadiation_acc   | Accumulated of accumulated solar radiation during PINF ( $W/m^2?$ )                  | 26 |             |
| NHoursSunlight_avg   | Average of number of hours of sunlight during PINF                                   | 27 |             |
| WindSpeed_avg        | Average of wind speed during PINF (Km/h)                                             | 22 |             |
| MaxWindSpeed_avg     | Average of maximum wind speed reached during PINF (Km/h)                             | 23 |             |
| MaxWindSpeed_max     | Maximum of maximum wind speed reached during PINF (Km/h)                             | 24 |             |
| Rain_avg             | Average of accumulated rain during PINF (mm)                                         | 15 | $\star$     |
| Rain_acc             | Accumulated of accumulated rain during PINF (mm)                                     | 13 | $\star$     |
| RelativeHumidity_avg | Average relative humidity during PINF (%)                                            | 7  | $\star$     |
| RelativeHumidity_max | Maximum relative humidity during PINF (%)                                            | 21 |             |
| NHRH95_avg           | Average number of hours with rel. humidity $> 95\%$ during PINF                      | 17 | $\star$     |
| NHRH95_max           | Maximum number of hours with rel. humidity $> 95\%$ during PINF                      | 19 |             |
| Tmed_HRH95_avg       | Average temperature of hours with rel. humidity $> 95\%$ during PINF ( $^{\circ}C$ ) | 7  | $\star$     |
| Tmed_HRH95_max       | Maximum temperature of hours with rel. humidity $> 95\%$ during PINF ( $^{\circ}C$ ) | 6  | $\star$     |
| NHNRH95_avg          | Average of hours during night with rel. humidity $> 95\%$ during PINF                | 14 | $\star$     |
| NHNRH95_max          | Maximum of hours during night with rel. humidity $> 95\%$ during PINF                | 9  | $\star$     |
| Tmed_NHNRH95_avg     | Average temp. of hours during night with rel. humidity $> 95\%$ during PINF          | 8  | $\star$     |
| Tmed_NHNRH95_max     | Maximum temp. of hours during night with rel. humidity $> 95\%$ during PINF          | 2  | $\star$     |

The full set of variables in our dataset are shown in Table 1, with the plot attribute variables and time related variables above, followed by the weather variables summarising conditions over the 45 days period of infection. We discretized all the non-discrete variables (other than the incidence) automatically, using equal frequency discretization with 3 bins by Weka implementation. We also obtained a reduced dataset, dataset $^r$ , applying Symmetrical Uncertainty (SU) with respect to the class<sup>4</sup> using cross-validation. Fig 4 shows the SU score for

<sup>4</sup> $SU(C, A_i) = 2 \cdot \frac{H(C) - H(C|A_i)}{H(C) + H(A_i)}$ , where  $C$  is the class,  $A_i$  an attribute and  $H()$  means entropy.

each variables, with our chosen cut point for the reduced dataset experiments (after the 17<sup>th</sup> variable). The variables left out are weather variables related to wind conditions and solar radiation, the year, the variables related to the days before and after the prediction, and somewhat surprisingly, the variables related to the load and spacing conditions of the plot.

## 5 BNs for predicting coffee rust

### 5.1 Base cases: handcrafted BNs

We handcrafted two simple BN models (see Fig.5), to use for baseline comparison. The first contains only two variables,  $Incidence_{d0} \rightarrow$

$Incidence_{target}$ , while in the second, we add the load and spacing variables.

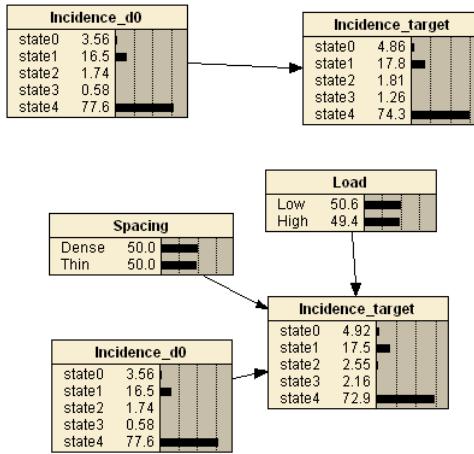


Figure 5: Simple BN handcrafted models

The parameters for both the handcrafted BNs were learnt (in Netica) with 80% of the available data. The other 20% was used for testing. The resultant confusion matrices are given in Table 2, showing error rates of 18.36% and 18.12%. Note that we combine the predictions into the following 3 categories (corresponding more closely to both the previous research and also the real world scenario):

- Negative: corresponds to States 0 and 1, 0-4% of leaves infected.
- Warning: corresponds to State 2, 4-4.5% of leaves infected.
- Positive: corresponds to States 3 and 4,  $\geq 4.5\%$  of leaves infected.

More interestingly, we also split the test data up into the cases where  $Incidence_{d0}$  is Negative or Warning and Positive. We can see that there is a very large difference between the predictions when coffee rust was previously present (well predicted, in most cases still present) and when it was negative/warning (much less well predicted). However the negative/warning cases are really those that matter, given that, in the real world, given the presence of coffee rust, fungicides would have been applied.

The error rates for the simplest handcrafted model are 37.33% when the model has been

Table 2: Confusion matrices for the simple handcrafted BNs.

| model  |   | all data |     | neg. inc. |    | pos. inc. |     |
|--------|---|----------|-----|-----------|----|-----------|-----|
|        |   | -        | +   | -         | +  | -         | +   |
| First  | - | 55       | 25  | 45        | 25 | 0         | 0   |
|        | w | 0        | 0   | 0         | 0  | 0         | 0   |
|        | + | 38       | 225 | 3         | 2  | 32        | 235 |
| Second | - | 56       | 25  | 49        | 26 | 0         | 0   |
|        | w | 0        | 0   | 0         | 0  | 0         | 0   |
|        | + | 37       | 225 | 0         | 0  | 32        | 235 |

learnt with the cases where the rust was not previously present, and 11.98% when  $Incidence_{d0}$  is Positive. In the second model, the error rates are 34.66% and 11.98% respectively.

## 5.2 BNs learnt by CaMML

We ran CaMML on both the full dataset (33 variables), and dataset<sup>r</sup> (17 variables, after variable selection, as described above in Sec. 4), with no structural priors, with 10-fold cross validation. We used the CaMML default of  $201n^3$  iterations (where n is the number of variables) in its search, and for each run tested with the “best” (lowest score) BN returned by CaMML. Fig. 6 shows one of the 10 BNs learned on reduced dataset<sup>r</sup>. The lack of prior information in the models leads to some relations that are not present in the real world, e.g. weather variables influencing the month instead of vice versa. However, the class variable has as parents the month and a humidity related variable, which seems reasonable. Curiously, the variable  $Incidence_{d0}$  is not directly linked to the target incidence.

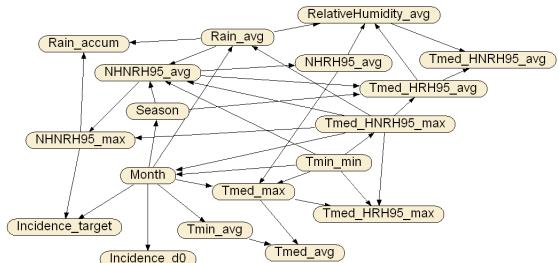


Figure 6: An example BN learned, without priors, from dataset<sup>r</sup>

We also ran CaMML (both full and reduced dataset) with a range of structural priors; here

(due to space) we describe only experiments on dataset<sup>r</sup>, with the following simple tiers<sup>5</sup>:

1.  $Incidence_{target}$  follows all other variables, using tiers ( $\{\text{All variables except } Inc_{target}\} \prec \{Incidence_{target}\}$ ), confidence 1.
2. *Month* and *Season* (the only non-weather variables in dataset<sup>r</sup>) come before all the weather variables, confidence 1.

Fig. 7 shows BNs learned on dataset<sup>r</sup>, using the first tier (above) and with both tiers (below). These models are similar to that learned without priors, where CaMML had  $Incidence_{target}$  as a leaf node. Note that now the parents of the class variable are the Season (itself a child of Month), a temperature related variable and  $Incidence_{d0}$  (first tier only); these seem more logical given existing domain knowledge.<sup>6</sup>

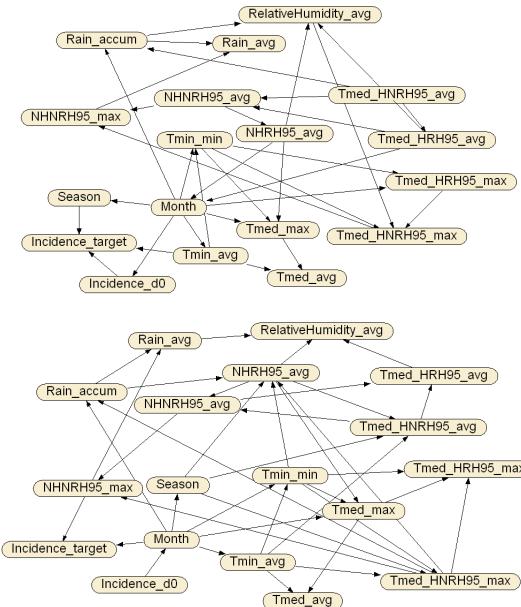


Figure 7: Example BNs learned from dataset<sup>r</sup> with first tier prior (above) and with both simple tiers (below).

Following the approach used in Sec. 5.1, Table 3 shows the confusion matrices obtained from the models learnt by CaMML with

<sup>5</sup>On the full dataset we used other types of priors, such as excluding possible connections between the background variables *Load*, *Spacing*, *Month* and *Year*.

<sup>6</sup>Although the arc from  $Incidence_{d0} \rightarrow Month$  with both tiers is undesirable; in our next iteration we'll add priors to make *Month* a root node.

dataset<sup>r</sup>. When learning without any prior information, the error rate is 9.84%. This rate decreases when learning with the first simple tier (9.32%) and with both tiers (8.97%).

Table 3: Confusion matrices for the models learned with dataset<sup>r</sup>.

|   |     | no tiers |     | simple tier |     | two tiers |   |
|---|-----|----------|-----|-------------|-----|-----------|---|
|   |     | -        | +   | -           | +   | -         | + |
| - | 330 | 77       | 315 | 52          | 298 | 28        |   |
| w | 2   | 0        | 1   | 0           | 0   | 0         |   |
| + | 92  | 1215     | 108 | 1240        | 126 | 1264      |   |

## 6 Comparison and evaluation

The errors obtained in (Luaces et al., 2011) range from 1.47% to 8.82% depending on the parameterization, the regressors show error rates from 4.41% to 8.82%, and the nondeterministic classifiers range from the 2.94%-6.47% of the LibSVM to LibLINEAR's 1.47%-5.29%.

To compare our results to other standard ML classifiers , we learnt using Weka (and evaluating with 10-fold cross validation) a Naive Bayes (NB) model, a decision tree (C4.5), using Tree Augmented Naive Bayes (TAN) and using Averaged one-dependence estimators (AODE). The obtained results are shown in Table 4; the most notable is clearly the low error rate obtained by the decision tree on the full dataset. The analysis of this decision tree is illuminating, as it shows that for the different states of the variable  $Incidence_{d0}$ , the variables that influence the decision change. The presence of these context-specific independencies (Boutilier et al., 1996) in the data partly explains why the BN learned models do not perform as well even with the presence of prior information.

Table 4: Error rates for the Weka classifiers.

|                      | NB    | C4.5 | TAN  | AODE |
|----------------------|-------|------|------|------|
| dataset              | 12.82 | 0.64 | 5.65 | 5.36 |
| dataset <sup>r</sup> | 13.9  | 8.3  | 7.63 | 7.16 |

Overall, our results using learned BNs are not that far from the previous studies, taking into account that those were tuned in their parametrizations and that this is the first study where BNs have been applied to this dataset.

The learned BN are better predictors than both the simple BNs (as expected) and NB, and comparable to AODE, TAN and C4.5 using dataset<sup>r</sup>. Finally, our results show how the incorporation of priors improves the performance of the model somewhat while producing more plausible causal structures.

## 7 Main conclusions and further work

We have described the novel application of probabilistic graphical models to coffee rust prediction, an agricultural case study. This process required a significant effort in analyzing the data and preprocessing it, as understanding the nature of the problem is the first step towards solving it. We applied causal learning (including with structural priors) to obtain a Bayesian network that predicts reasonably well (including outperforming NB), although it does not outperform the previous approaches yet. We have discovered through the learning of a decision tree how the influence of certain variables change for different contexts of  $Incidence_{d0}$ , suggesting further analysis of this is warranted.

An extended investigation of different methods for variable selection and discretization, as well as trying different combinations of priors, may improve prediction results. We also plan to apply pre-processing techniques for imbalanced datasets, and look at some of the recent work on characterising classification problem complexity (which is known to affect various methods differently). In the end, improved performance may be not be possible if there just isn't enough data to learn such a complex graphical model, especially for the case with incidence  $< 4.5\%$  the previous month.

Apart from doing further ML experimentation, the future work includes obtaining feedback from domain experts about the structures learned. By combining BNs with cost matrices, we aim to produce a decision network that will allow farmers to explore the tradeoffs when deciding whether to apply fungicides.

## Acknowledgments

This work has been partially funded by FEDER funds and the Spanish Government (MICINN) through projects TIN2010-20900-C04-01, 03 and the FPI schol-

arship programme (BES-2008-002049). We thank Luiz Henrique Rodrigues for providing the coffee rust dataset and answering our queries about it, and Steven Mascaro for assisting with preliminary data analysis.

## References

- P. A. Aguilera, A. Fernández, R. Fernández, R. Rumí, and A. Salmerón. 2011. Bayesian networks in environmental modelling. *Environmental Modelling Software*, 26(12):1376–1388.
- J. Avelino, H. Zelaya, A. Merlo, A. Pineda, M. Ordoñez, and S. Savary. 2006. The intensity of a coffee rust epidemic is dependent on production situations. *Ecological Modelling*, 197(34):431 – 447.
- C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. 1996. Context-specific independence in bayesian networks. In *UAI*, pages 115–123.
- A. L. Jensen. 1995. *A probabilistic model based decision support system for mildew management in winter wheat*. Ph.D. thesis, Aalborg University.
- K. B. Korb and A. E. Nicholson. 2010. *Bayesian Artificial Intelligence*. Chapman & Hall/CRC, Boca Raton, second edition.
- K. Kristensen and I. A. Rasmussen. 2002. The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture*, 33(3):197–217.
- A. C. Kushalappa. 1990. Development of forecasts: Timing fungicide applications to manage coffee rust and carrot blight. *Canadian Journal of Plant Pathology*, 12(1):92–99.
- O. Luaces, L. H. A. Rodrigues, C. A. A. Meira, J. R. Quevedo, and A. Bahamonde. 2010. Viability of an alarm predictor for coffee rust disease using interval regression. In *IEA/AIE (2)*, pages 337–346.
- O. Luaces, L. H. A. Rodrigues, C. A. A. Meira, and A. Bahamonde. 2011. Using nondeterministic learners to alert on coffee rust disease. *Expert Syst. Appl.*, 38(11):14276–14283.
- C. A. A. Meira, L. H. A. Rodrigues, and S. A. Moraes. 2008. Análise da epidemia da ferrugem do cafeiro com árvore de decisão. *Tropical Plant Pathology*, 33:114–124.
- R. O'Donnell, A. E. Nicholson, B. Han, K. B. Korb, M. J. Alam, and L. Hope. 2006. Causal discovery with prior information. In A. Sattar and B. H. Kang, editors, *AI 2006: Advances in Artificial Intelligence*, LNAI Series, pages 1162–1167. Springer-Verlag, Germany.
- C. S. Wallace. 2005. *Statistical and Inductive Inference by Minimum Message Length*. Springer, Berlin, Germany.
- I. H. Witten and E. Frank. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, third edition.

# Generalised Co-variation for Sensitivity Analysis in Bayesian Networks

Silja Renooij

Utrecht University, The Netherlands

S.Renooij@uu.nl

## Abstract

Upon varying parameters in a sensitivity analysis of a Bayesian network, the standard approach is to co-vary the parameters from the same conditional distribution such that their proportions remain the same. Alternative co-variation schemes are, however, possible. We theoretically investigate the effects of using alternative co-variation schemes on the so-called sensitivity function, and conclude that its general form remains the same under any linear co-variation scheme. In addition, we generalise the CD-distance for bounding global belief change, and prove a tight lower bound on this distance for parameter changes in single conditional probability tables.

## 1 Introduction

Sensitivity analysis is a general technique for studying the effects of parameter changes on the output of a mathematical model. In the context of Bayesian networks the effect of changes, applied to one or more probabilities from the network's conditional probability tables, on computed probabilities is determined. The results can be captured in detail by means of a *sensitivity function*, describing an output probability of interest in terms of one or more parameter probabilities. More global effects can be described by the *CD-distance*, which is a measure for bounding probabilistic belief change and complements the sensitivity function by giving insight in the effect of parameter changes on the global joint distribution, rather than on a specific (posterior) output probability of interest.

Upon varying a probability from a conditional distribution, the remaining probabilities from the same distribution need to be co-varied. The *proportional scheme* has been adopted as the standard scheme for co-variation in Bayesian networks, and various sensitivity analysis algorithms build upon this scheme. The proportional co-variation scheme is one of numerous alternatives for co-varying parameters from the same distribution. The mere fact that it is the standard co-variation scheme used, does not imply that there are no situations in which alternative schemes are suitable. However, the known standard form of the sensitivity function is based

on proportional co-variation, and the proportional scheme is known to be optimal when varying a single parameter, in the sense that it minimises the CD-distance (Chan & Darwiche, 2005).

It is as of yet unknown if the proportional scheme is optimal when multiple, independent parameters from are varied. Moreover, we may not be interested in minimising the CD-distance: for example, we may be interested in minimising KL-divergence, which is not equivalent to minimising CD-distance (Chan & Darwiche, 2005); or we may want to perform our analyses in the context of large disturbances, rather than minimal ones. In this paper we will therefore investigate exactly how both the sensitivity function and the CD-distance depend on the co-variation scheme used. We show that the general form of the sensitivity function is maintained as long as the co-variation scheme is linear in the parameter(s) varied. In addition, we generalise the CD-distance to arbitrary co-variation schemes, and prove that a previously suggested approximation of this distance is in fact a lower bound.

This paper is organised as follows. Section 2 provides preliminaries on sensitivity analysis and co-variation. Section 3 generalises the sensitivity function to arbitrary co-variation schemes; the consequences for computing the functions are discussed in Section 4. Section 5 likewise generalises the CD-distance. The paper ends with conclusions and directions for future research in Section 6.

## 2 Preliminaries

A Bayesian network compactly represents a joint probability distribution  $\text{Pr}$  over a set of stochastic variables  $\mathbf{V}$  (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  $\Theta_{V_i|\pi(V_i)}$  for each variable  $V_i$  and its parents  $\pi(V_i)$  in the graph, such that  $\text{Pr}(\mathbf{V}) = \prod_i \Theta_{V_i|\pi(V_i)}$ . We will refer to  $\Theta_{V_i|\pi(V_i)}$  as the conditional probability table (CPT) of  $V_i$ ; entries  $\theta$  of  $\Theta$  are called parameter probabilities, or parameters for short. 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 output probabilities of interest are studied. The effects of such  $n$ -way parameter variation can be described by *sensitivity functions*. Such a function is *multilinear* in the varied parameters in case of a prior probability of interest, and *rational* (quotient of two multilinear functions) in the posterior case (Coupé & Van der Gaag, 2002). For example, the 1-way sensitivity function  $f_a^{\mathbf{e}}(x)$  describing the posterior probability  $\text{Pr}(a | \mathbf{e})$  as a function of parameter  $x$  is given by

$$f_a^{\mathbf{e}}(x) = \frac{f_{a,\mathbf{e}}(x)}{f_{\mathbf{e}}(x)} = \frac{c_1 x + c_0}{d_1 x + d_0} \quad (1)$$

with constants  $c_i, d_i, i = 0, 1$ , built from non-varied network parameters. The general form of the sensitivity function was established under the assumption of *proportional* co-variation (Castillo, Gutiérrez & Hadi, 1997; Coupé & Van der Gaag, 2002).

**Co-variation** Consider a binary-valued variable  $V$  with values  $v$  and  $\bar{v}$ , and parent configuration  $\mathbf{u}$ . Since  $\theta_{v|\mathbf{u}} + \theta_{\bar{v}|\mathbf{u}} = 1$ , we have that if  $\theta_{v|\mathbf{u}}$  varies,  $\theta_{\bar{v}|\mathbf{u}}$  should be co-varied to ensure that their sum remains 1. Therefore, if  $\theta_{v|\mathbf{u}}$  varies as  $x$  in a sensitivity analysis,  $\theta_{\bar{v}|\mathbf{u}}$  should co-vary as  $1 - x$ .

Now if  $V$  has  $t > 2$  values  $v_1, \dots, v_t$ , and  $x = \theta_{v_1|\mathbf{u}}$  is the parameter varied in a sensitivity analysis, then there are endless ways in which the parameters  $\theta_{v_k|\mathbf{u}}, 1 < k \leq t$  can co-vary with  $x$ . Using the

above mentioned proportional co-variation scheme, the parameters  $\theta_{v_k|\mathbf{u}}, k \neq 1$ , get the same proportion of the remaining mass of  $1 - x$ , as they had originally:

$$\theta_{v_k|\mathbf{u}}^* = \frac{\theta_{v_k|\mathbf{u}}}{1 - \theta_{v_1|\mathbf{u}}} \cdot (1 - x)$$

where  $\theta_{v_k|\mathbf{u}}^*$  is the new value of the parameter, and  $\theta_{v_k|\mathbf{u}}, \theta_{v_1|\mathbf{u}}$  indicate the original values that were specified in the network. We typically assume that parameters with an original value of 0 or 1 are not varied, so the above denominator is in  $\langle 0, 1 \rangle$ .

The proportional scheme has been adopted as the standard scheme for co-variation in Bayesian networks, and various sensitivity analysis algorithms build upon this scheme (Chan & Darwiche (2002; 2004; 2005), Coupé & Van der Gaag (2002), Kjærulff & Van der Gaag (2000)). In fact, the proportional scheme minimises the CD-distance between the new distribution  $\text{Pr}^*$  and the original distribution  $\text{Pr}$  (Chan & Darwiche, 2005).

## 3 Co-variation in the Sensitivity Function

The proportional co-variation scheme, although standard, is merely one of many alternatives for co-varying parameters from the same distribution. In this section, we will take a fresh look at sensitivity functions without restricting ourselves to a particular co-variation scheme.

### 3.1 Co-variation schemes

Consider a  $t$ -valued variable  $V$  and suppose we vary a parameter from the distribution  $\Theta_{V|\mathbf{u}}$  as  $x$ . The  $t - 1$  remaining parameters from this distribution must co-vary; more specifically, each of these parameters should get a portion, or cut, of the remaining mass  $1 - x$ . We define a valid co-variation scheme based on these cuts.

**Definition 1.** Consider  $k \geq 1$  parameters  $\theta_k$  from the same distribution and let  $m \leq 1$  be the total probability mass available for these parameters. A *co-variation cut*  $\gamma : \{\theta_k\} \rightarrow [0, 1]$  defines the share of each parameter  $\theta_k$  in  $m$ . A *co-variation scheme*  $s(k) = \gamma(k) \cdot m$  now maps each  $\theta_k$  to a new value  $\theta_k^*$ . A co-variation scheme is called *valid* if  $\sum_k \gamma(k) = 1$ .

From here on we will write  $\gamma_k$  rather than  $\gamma(k)$ . Note that a valid co-variation scheme ensures that the entire distribution under consideration sums to  $(1 - m) + \sum_k \gamma_k \cdot m = 1$ . The following example gives two valid co-variation schemes.

**Example 1.** The standard proportional co-variation scheme,

$$\gamma_{v_k|u} \cdot (1 - x) = \frac{\theta_{v_k|u}}{1 - \theta_{v_1|u}} \cdot (1 - x) \quad (2)$$

is indeed valid:

$$\sum_{k \neq 1} \gamma_{v_k|u} = \sum_{k=2}^t \frac{\theta_{v_k|u}}{1 - \theta_{v_1|u}} = \frac{1 - \theta_{v_1|u}}{1 - \theta_{v_1|u}} = 1$$

We can also think of other valid co-variation schemes. Consider for example a *uniform* co-variation scheme:

$$\gamma_{v_k|u} \cdot (1 - x) = \frac{1}{t - 1} \cdot (1 - x) \quad (3)$$

which uniformly distributes the remaining mass of  $1 - x$  over the  $t - 1$  co-varying parameters. This scheme is also valid:

$$\sum_{k \neq 1} \gamma_{v_k|u} = \sum_{k=2}^t \frac{1}{t - 1} = (t - 1) \cdot \frac{1}{1 - t} = 1 \quad \blacksquare$$

### 3.2 The generalised 1-way form

In the remainder of this paper we focus on a probability of interest  $\Pr(\mathbf{w})$ <sup>1</sup> as a function of a parameter  $\theta_{v_1|u}$  of a  $t$ -valued variable  $V$ . In addition, without loss of generality, we often assume  $V$  to have a single, binary-valued parent  $U$  with values  $u$  and  $\bar{u}$ .

The following proposition explicitly captures how the general form of the 1-way sensitivity function depends on the co-variation scheme.

**Proposition 1.** *Probability  $\Pr(\mathbf{w})$  as a function of  $x = \theta_{v_1|u}$  of  $t$ -valued variable  $V$  is given by*

$$f_{\mathbf{w}}(x) = (\alpha - \beta^\gamma) \cdot x + (\beta^\gamma + \delta)$$

<sup>1</sup>Note from Equation 1 that  $\Pr(\mathbf{w})$  is general enough to represent any probability of interest. For example,  $\Pr(a | \mathbf{e}) = \Pr(a, \mathbf{e}) / \Pr(\mathbf{e})$ , where both numerator and denominator are of the form  $\Pr(\mathbf{w})$ .

where

$$\begin{aligned} \alpha &= \Pr(\mathbf{w}|v_1, u) \cdot \Pr(u), \delta = \Pr(\mathbf{w}, \bar{u}), \\ \beta^\gamma &= \sum_{k=2}^t \Pr(\mathbf{w}|v_k, u) \cdot \Pr(u) \cdot \gamma_{v_k|u}, \end{aligned}$$

and  $\gamma_{v_k|u} \cdot (1 - x)$ , for all  $\theta_{v_k|u}$ ,  $1 < k \leq t$ , is a co-variation scheme.

*Proof.* First we rewrite  $\Pr(\mathbf{w})$  to include the parameters under consideration:

$$\begin{aligned} \Pr(\mathbf{w}) &= \Pr(\mathbf{w}, \bar{u}) + \Pr(\mathbf{w}, u) \\ &= \Pr(\mathbf{w}, \bar{u}) + \Pr(\mathbf{w}|v_1, u) \cdot \Pr(u) \cdot \theta_{v_1|u} \\ &\quad + \sum_{k=2}^t \Pr(\mathbf{w}|v_k, u) \cdot \Pr(u) \cdot \theta_{v_k|u} \end{aligned}$$

The proposition now follows by replacing  $\theta_{v_1|u}$  by  $x$  and each  $\theta_{v_k|u}$  by its value according to the co-variation scheme.  $\square$

### 3.3 The generalised $n$ -way form

In this section we consider the explicit simultaneous variation of  $n > 1$  parameters from either a single distribution, or from an entire CPT. Taking parameters from multiple CPTs is also possible (Chan & Darwiche, 2004; Kjærulff & Van der Gaag, 2000). Although our results extend to this latter case, it will not be considered here: the approach is not often used in practice since it quickly becomes computationally infeasible.

The following proposition explicitly captures how the general form of the  $n$ -way sensitivity function for  $n$  parameters from a single distribution  $\Theta_{V|u}$ , depends on the co-variation scheme.

**Proposition 2.** *Probability  $\Pr(\mathbf{w})$  as a function of  $n$  parameters  $x_i = \theta_{v_i|u}$ ,  $1 \leq i \leq n$ , of  $t$ -valued variable  $V$ ,  $t > n$ , is given by*

$$f_{\mathbf{w}}(x_1, \dots, x_n) = \sum_{i=1}^n (\alpha_i - \beta^\gamma) \cdot x_i + (\beta^\gamma + \delta)$$

where

$$\begin{aligned} \alpha_i &= \Pr(\mathbf{w}|v_i, u) \cdot \Pr(u), \delta = \Pr(\mathbf{w}, \bar{u}), \\ \beta^\gamma &= \sum_{k=n+1}^t \Pr(\mathbf{w}|v_k, u) \cdot \Pr(u) \cdot \gamma_{v_k|u}, \end{aligned}$$

and  $\gamma_{v_k|u} \cdot (1 - \sum_{i=1}^n x_i)$ , for all  $\theta_{v_k|u}$ ,  $n < k \leq t$ , is a co-variation scheme.

*Proof.* The proof is analogous to that of Proposition 1, with each  $\theta_{v_i|u}$ ,  $1 \leq i \leq n$ , replaced by  $x_i$  and each  $\theta_{v_k|u}$ ,  $n < k \leq t$ , replaced by its value according to the co-variation scheme.  $\square$

Note that in the above case we need to ensure that  $\sum_{i=1}^n x_i \leq 1$ , which means we have to impose an additional constraint on the sensitivity analysis. For this reason, an  $n$ -way analysis typically considers parameters from  $n$  different distributions, often constituting an entire CPT. That is, from a single CPT  $\Theta_{V|U}$ , exactly one parameter for each conditional distribution  $\Theta_{V|U_j}$  is varied, and all other parameters are co-varied (Chan & Darwiche, 2004). For such single-CPT  $n$ -way analyses, the following proposition describes how the general form of the sensitivity function depends on the co-variation scheme.

**Proposition 3.** *Probability  $\Pr(\mathbf{w})$  as a function of  $n$  parameters  $x_j = \theta_{v_1|\mathbf{u}_j}$ ,  $1 \leq j \leq n$ , of  $t$ -valued variable  $V$ , is given by*

$$f_{\mathbf{w}}(x_1, \dots, x_n) = \sum_{j=1}^n (\alpha_j - \beta_j^\gamma) \cdot x_j + \beta_j^\gamma$$

where

$$\begin{aligned} \alpha_j &= \Pr(\mathbf{w}|v_1, \mathbf{u}_j) \cdot \Pr(\mathbf{u}_j), \\ \beta_j^\gamma &= \sum_{k=2}^t \Pr(\mathbf{w}|v_k, \mathbf{u}_j) \cdot \Pr(\mathbf{u}_j) \cdot \gamma_{v_k|\mathbf{u}_j}, \end{aligned}$$

and  $\gamma_{v_k|\mathbf{u}_j} \cdot (1 - x_j)$ , for all  $\theta_{v_k|\mathbf{u}_j}$ ,  $1 < k \leq t$ , are co-variation schemes.

*Proof.* The proof is analogous to that of Proposition 1, except that for each parent configuration  $\mathbf{u}_j$ , the term  $\Pr(\mathbf{w}, \mathbf{u}_j)$  now depends on an  $x_j$  and its co-varying parameters.  $\square$

### 3.4 Fixing parameters

In the above, we assumed that we varied one or more parameters from a  $t$ -valued variable and let the remaining parameters co-vary in some way. We may, however, want one or more parameters to stick to their original value. For example, parameters with an original value of zero indicate an impossibility that should remain impossible.

We now consider the effect of fixing parameters on the sensitivity function. Without loss of generality we assume that upon varying parameter  $\theta_{v_1|\mathbf{u}}$

of a  $t$ -valued variable  $V$ ,  $t \geq 3$ , a single parameter  $\theta_{v_t|\mathbf{u}}$  should remain unchanged.

**Proposition 4.** *Probability  $\Pr(\mathbf{w})$  as a function of  $x = \theta_{v_1|u}$  of  $t$ -valued variable  $V$ ,  $t \geq 3$ , with parameter  $\theta_{v_t|u}$  fixed, is given by*

$$f_{\mathbf{w}}(x) = (\alpha - \beta^\gamma) \cdot x + (\mu \cdot \beta^\gamma + \delta)$$

where

$$\begin{aligned} \alpha &= \Pr(\mathbf{w}|v_1, u) \cdot \Pr(u), \\ \delta &= \Pr(\mathbf{w}|v_t, u) \cdot \Pr(u) \cdot \theta_{v_t|u} + \Pr(\mathbf{w}, \bar{u}), \\ \beta^\gamma &= \sum_{k=2}^{t-1} \Pr(\mathbf{w}|v_k, u) \cdot \Pr(u) \cdot \gamma_{v_k|u}, \\ \mu &= 1 - \theta_{v_t|u} \text{ (mass for co-variation)}, \end{aligned}$$

and  $\gamma_{v_k|\mathbf{u}} \cdot (\mu - x)$ , for all  $\theta_{v_k|\mathbf{u}}$ ,  $1 < k < t$ , is a co-variation scheme.

*Proof.* Reconsider the proof of Proposition 1. The proposition follows directly by taking into account that we no longer co-vary all  $t - 1$  remaining parameters, and that the co-varied parameters together only have a remaining mass of  $(1 - \theta_{v_t|u}) - x = \mu - x$  to divide.  $\square$

Proposition 4 generalises to any fixed mass  $1 - \mu$ , including the special case  $\mu = 1$  where no parameters are fixed, as is the standard case. Note that fixing parameters with an original value of zero also gives  $\mu = 1$ , so we may have to exclude them explicitly from the co-variation. This is the case when using, for example, the uniform co-variation scheme. Using proportional co-variation, however, parameters with an original value of zero keep that value upon co-variation:  $\frac{0}{1-\theta_{v_1|u}}(1-x) = 0$ . Proposition 4 also generalises to the  $n$ -way functions considered in Propositions 2 and 3.

### 3.5 Generalised versus standard functions

We note that in the propositions above, the term  $\beta^\gamma$  (or  $\beta_j^\gamma$ ), sums over all the co-varying parameters. It is exactly this term that depends on the co-variation scheme used. The propositions therefore explicitly indicate which part of the sensitivity function depends on the co-variation scheme used. We now note that  $\beta^\gamma$  can be considered constant with respect to the varied parameter(s)  $x$ , only if  $\gamma_{v_k|\mathbf{u}}$  is independent of  $x$ , i.e. if the co-variation scheme  $\gamma_{v_k|\mathbf{u}} \cdot (1 - x)$  is linear in  $x$ .

**Corollary 1.** A sensitivity function  $f_a^e(x)$  is of the standard form (Equation (1)), iff the co-variation scheme used is valid and linear in  $x$ .

*Proof.* The result follows directly from Definition 1 and Proposition 1.  $\square$

The above corollary generalises to  $n$ -way sensitivity functions, as well as to sensitivity functions with fixed parameters. From the corollary we have that, contrary to the usual assumption in literature, the general form of the sensitivity function does not necessarily require a proportional co-variation scheme: any valid scheme, linear in  $x$  will do.

**Example 2.** The uniform co-variation scheme (Equation (3)) is clearly linear in  $x$ . The standard proportional co-variation scheme (Equation (2)) is also linear in  $x$ , since  $\gamma_{v_k|u}$  only depends on the *original* value of  $\theta_{v_1|u}$ .  $\blacksquare$

## 4 Using Another Co-variation Scheme

In the previous section we concluded that sensitivity functions keep their standard form as long as a valid and linear co-variation scheme is used. We now investigate the implications of this for existing algorithms that establish sensitivity functions, and provide a preliminary comparison of such functions for alternative co-variation schemes.

### 4.1 Computing the constants

Algorithms that build upon the analytic expression of the constants of the sensitivity function for their computation, such as e.g. Kjærulff & Van der Gaag (2000), cannot be applied if we use a co-variation scheme that is different from proportional co-variation. This is simply because the co-variation scheme affects the analytic form of the constants, as we have demonstrated in the previous section.

Another class of algorithms constructs and solves a system of  $r$  equations, one for each required constant, by computing the output probability of interest for  $r$  different values of  $x$  (see e.g. Coupé & Van der Gaag (2002)). These algorithms do not depend on the analytic form of the constants, and can therefore be applied regardless of the co-variation scheme used.

**A note on uniform co-variation** As illustrated by the following example, uniform co-variation can be implemented by applying proportional co-variation to a pre-processed CPT. Hence, algorithms that build on the analytic form of the constants *can* be applied in the context of uniform co-variation.

**Example 3.** Consider parameters  $\theta_{v_1|u}$ ,  $\theta_{v_2|u}$  and  $\theta_{v_3|u}$  with values 0.3, 0.5 and 0.2, respectively. Suppose we vary  $\theta_{v_1|u}$  to 0.5 and co-vary the other two: under uniform co-variation this results in a value of 0.25 for both. The same result is achieved by using proportional co-variation, *after* first uniformly distributing  $(1 - \theta_{v_1|u})$  over  $\theta_{v_2|u}$  and  $\theta_{v_3|u}$ . This approach is valid, since equal parameter values remain equal under proportional co-variation.  $\blacksquare$

### 4.2 Comparing co-variation schemes

In this section we provide a preliminary analysis of the impact of using different co-variation schemes. Example 3 points to a potential problem with alternative co-variation schemes, especially those independent of the original value of  $\theta_{v_1|u}$ : if a parameter  $x$ , upon variation, takes on a value that corresponds to the original value  $x^0$  of the parameter, then the co-varying parameters may take on different values from their original ones. As a result, for a probability of interest  $\Pr(a|e)$  with original value  $p^0$  we will find that  $f_a^e(x^0) \neq p^0$ . This may seem counter-intuitive, but the sensitivity function describes the relation between an output probability and the value of parameter  $x$ , *in the context of the values for the co-varied parameters*. Upon choosing a uniform co-variation scheme, for example,  $f(x^0)$  is computed assuming that the co-varying parameters are distributed uniformly over  $1 - x^0$ , even if they originally weren't. Note that for the proportional co-variation scheme, we indeed have that  $\gamma_{v_k|u} \cdot (1 - x^0) = \theta_{v_k|u} \forall k$ . From the above observations, we have that properties that assume the sensitivity function to include  $(x^0, p^0)$  may give corrupt results upon applying alternative co-variation schemes; this includes properties such as the sensitivity value and various bounds on the sensitivity function (Chan & Darwiche, 2002; Van der Gaag, Renooij & Coupé, 2007).

With respect to sensitivity functions which are defined for  $x \in [0, 1]$ , i.e. none of the co-varying

parameters are fixed to a non-zero value, we observe the following:

- if a parameter is varied to  $x = 1$ , there is no more mass left for co-variation, so  $f_w(1)$  is independent of the scheme used; this also holds for rational functions, and for  $n$ -way functions.
- for 1-way linear sensitivity functions we have that for any co-variation scheme which preserves  $p^0$  for  $x = x^0$ , the function is independent of the co-variation scheme. This observation, however, does not necessarily hold for 1-way rational functions, which are uniquely defined by three points.
- the maximum difference in 1-way linear sensitivity functions, caused by using different co-variation schemes, is thus found for  $x = 0$ ; whether the same can be said for 1-way rational functions requires further investigation.

From these observations we conjecture that considering alternative co-variation schemes may be most interesting for parameters with smaller values. In the context of parameter tuning, it could turn out that a 'single' parameter change under an alternative co-variation scheme can accomplish an effect that is not possible with a 'single' parameter change under proportional co-variation.

## 5 Co-variation and the CD-distance

The CD-distance measures the distance between two probability distributions  $\text{Pr}$  and  $\text{Pr}^*$ . If  $\text{Pr}^*$  is the result of making changes in a single CPT  $\Theta_{V|U}$  in a Bayesian network, the CD-distance is given by (Chan & Darwiche, 2004):

$$D(\Theta_{V|U}, \Theta_{V|U}^*) = \ln \max_{v_i, u_j} \frac{\theta_{v_i|u_j}^*}{\theta_{v_i|u_j}} - \ln \min_{v_i, u_j} \frac{\theta_{v_i|u_j}^*}{\theta_{v_i|u_j}}$$

In this section we will analyse how the CD-distance depends on the co-variation scheme used.

### 5.1 Single parameter co-variation

Chan & Darwiche (2005) demonstrated that upon changing a single parameter  $\theta_{v_1|u}$  from a distribution  $\Theta_{V|u}$ , the proportional co-variation scheme is optimal in the sense that it minimises the CD-distance  $D$  between the original distribution  $\Theta_{V|u}$

and the new distribution  $\Theta_{V|u}^*$ . In addition, they showed that this distance has the following closed form:<sup>2</sup>

$$D_p(\Theta_{V|u}, \Theta_{V|u}^*) = \left| \ln \frac{\theta_{v_1|u}^*}{\theta_{v_1|u}} - \ln \frac{1 - \theta_{v_1|u}^*}{1 - \theta_{v_1|u}} \right| \quad (4)$$

Opting for a different co-variation scheme will therefore necessarily increase the distance between original and new distribution. It is unknown, however, how the CD-distance is exactly affected by the co-variation scheme used. The following proposition provides us with that information.

**Proposition 5.** Consider changing a parameter  $\theta_{v_1|u}$  for a  $t$ -valued variable,  $t \geq 2$ , to  $\theta_{v_1|u}^*$ . Let  $\gamma_{v_k|u} \cdot (1 - \theta_{v_1|u}^*)$ ,  $1 < k \leq t$ , be the new values of the co-varied parameters. Then,  $D_\gamma(\Theta_{V|u}, \Theta_{V|u}^*) =$

$$= \ln \max \left\{ \frac{\theta_{v_1|u}^*}{\theta_{v_1|u}}, \frac{1 - \theta_{v_1|u}^*}{\min_{1 < k \leq t} \gamma_{v_k|u}^{-1} \cdot \theta_{v_k|u}} \right\} - \ln \min \left\{ \frac{\theta_{v_1|u}^*}{\theta_{v_1|u}}, \frac{1 - \theta_{v_1|u}^*}{\max_{1 < k \leq t} \gamma_{v_k|u}^{-1} \cdot \theta_{v_k|u}} \right\}$$

*Proof.* For the CD-distance we need to compute

$$\ln \max_{1 \leq i \leq t} \frac{\theta_{v_i|u}^*}{\theta_{v_i|u}} - \ln \min_{1 \leq i \leq t} \frac{\theta_{v_i|u}^*}{\theta_{v_i|u}}$$

We therefore consider all ratios  $\theta_{v_i|u}^*/\theta_{v_i|u}$ ,  $1 \leq i \leq t$  and determine their maximum and minimum, respectively. For  $i \neq 1$ , we have that

$$\frac{\theta_{v_i|u}^*}{\theta_{v_i|u}} = \frac{\gamma_{v_i|u} \cdot (1 - \theta_{v_1|u}^*)}{\theta_{v_i|u}} = \frac{1 - \theta_{v_1|u}^*}{\gamma_{v_i|u}^{-1} \cdot \theta_{v_i|u}}$$

It is obvious that

$$\max_{1 < k \leq t} \frac{1 - \theta_{v_1|u}^*}{\gamma_{v_k|u}^{-1} \cdot \theta_{v_k|u}} = \frac{1 - \theta_{v_1|u}^*}{\min_{1 < k \leq t} \gamma_{v_k|u}^{-1} \cdot \theta_{v_k|u}}$$

and

$$\min_{1 < k \leq t} \frac{1 - \theta_{v_1|u}^*}{\gamma_{v_k|u}^{-1} \cdot \theta_{v_k|u}} = \frac{1 - \theta_{v_1|u}^*}{\max_{1 < k \leq t} \gamma_{v_k|u}^{-1} \cdot \theta_{v_k|u}}$$

<sup>2</sup>The subscript for  $D$  indicates the assumed co-variation scheme:  $\gamma$  is general,  $p$  is proportional,  $u$  is uniform.

Comparing the above minimum and maximum to the ratio  $\theta_{v_1|\mathbf{u}}^*/\theta_{v_1|\mathbf{u}}$  for  $i = 1$ , we straightforwardly find that there exist values for  $\theta_{v_k|\mathbf{u}}$  such that  $\theta_{v_1|\mathbf{u}}^*/\theta_{v_1|\mathbf{u}}$  is the largest term, but there are also values for  $\theta_{v_k|\mathbf{u}}$  such that  $\theta_{v_1|\mathbf{u}}^*/\theta_{v_1|\mathbf{u}}$  is the smallest term (see example below). The result follows from these observations.  $\square$

The following example illustrates the various possibilities.

**Example 4.** Consider a distribution over a variable  $V$  with 3 values. Suppose we vary parameter  $\theta_{v_1|\mathbf{u}}$  and co-vary the remaining two using the *uniform* co-variation scheme. Let the original assessments for  $v_1$ ,  $v_2$  and  $v_3$  be 0.25, 0.7 and 0.05, respectively. Then  $\gamma_{v_2|\mathbf{u}}^{-1} \cdot \theta_{v_2|\mathbf{u}} = (3-1) \cdot 0.7 = 1.4$  and  $\gamma_{v_3|\mathbf{u}}^{-1} \cdot \theta_{v_3|\mathbf{u}} = (3-1) \cdot 0.05 = 0.1$ .

**Case 1** ( $\max = \theta_{v_1|\mathbf{u}}/\theta_{v_1|\mathbf{u}}^*$ ): we vary  $\theta_{v_1|\mathbf{u}}$  = 0.25 to  $\theta_{v_1|\mathbf{u}}^* = 0.8$ ; then  $\theta_{v_1|\mathbf{u}}^*/\theta_{v_1|\mathbf{u}} = 0.8/0.25 = 3.2$ . Therefore  $D_u(\Theta_{V|\mathbf{u}}, \Theta_{V|\mathbf{u}}^*) = \ln \max\{3.2, 0.2/0.1\} - \ln \min\{3.2, 0.2/1.4\} = 3.109$ .

**Case 2** ( $\max \geq \theta_{v_1|\mathbf{u}}/\theta_{v_1|\mathbf{u}}^* \geq \min$ ): we vary  $\theta_{v_1|\mathbf{u}}$  to  $\theta_{v_1|\mathbf{u}}^* = 0.5$ ; then  $\theta_{v_1|\mathbf{u}}^*/\theta_{v_1|\mathbf{u}} = 0.5/0.25 = 2$ . Therefore  $D_u(\Theta_{V|\mathbf{u}}, \Theta_{V|\mathbf{u}}^*) = \ln \max\{2, 0.5/0.1\} - \ln \min\{2, 0.5/1.4\} = 2.639$ .

**Case 3** ( $\min = \theta_{v_1|\mathbf{u}}/\theta_{v_1|\mathbf{u}}^*$ ): we vary  $\theta_{v_1|\mathbf{u}}$  to 0.10; then  $\theta_{v_1|\mathbf{u}}^*/\theta_{v_1|\mathbf{u}} = 0.1/0.25 = 0.4$ . Therefore  $D_u(\Theta_{V|\mathbf{u}}, \Theta_{V|\mathbf{u}}^*) = \ln \max\{0.4, 0.9/0.1\} - \ln \min\{0.4, 0.9/1.4\} = 3.114$ .

Note that if we use the *proportional* co-variation scheme, then  $\gamma_{v_2|\mathbf{u}}^{-1} \cdot \theta_{v_2|\mathbf{u}} = (0.7/0.75)^{-1} \cdot 0.7 = 0.75$  and  $\gamma_{v_3|\mathbf{u}}^{-1} \cdot \theta_{v_3|\mathbf{u}} = (0.05/0.75)^{-1} \cdot 0.05 = 0.75$ , which indeed both equal  $1 - \theta_{v_1|\mathbf{u}}$ , as in Equation (4). For **case 1**, for example, we then get  $D_p(\Theta_{V|\mathbf{u}}, \Theta_{V|\mathbf{u}}^*) = \ln \max\{3.2, 0.2/0.75\} - \ln \min\{3.2, 0.2/0.75\} = 2.485$  which is indeed less than the distance found with uniform co-variation.  $\blacksquare$

Note from the above example that the minimisation and maximisation terms over the co-varied parameters depend on  $\gamma_{v_k|\mathbf{u}}$ , but not on  $\theta_{v_1|\mathbf{u}}^*$ ; hence they can be pre-computed and used for computing the CD-distance for any new distribution  $\Theta_{V|\mathbf{u}}^*$  obtained by changing parameter  $\theta_{v_1|\mathbf{u}}$  to  $\theta_{v_1|\mathbf{u}}^*$ .

## 5.2 Single CPT co-variation

Chan & Darwiche (2004) erroneously introduced the following closed form for the CD-distance in case the parameters of an entire CPT  $\Theta_{V|\mathbf{U}}$  are varied as described in Section 3.3 (just above Proposition 3), i.e. a single parameter from *each*  $\Theta_{V|\mathbf{u}_j}$  is varied, while co-varying all others:

$$\max_{\mathbf{u}_j} \left| \ln \frac{\theta_{v_1|\mathbf{u}_j}^*}{\theta_{v_1|\mathbf{u}_j}} - \ln \frac{1 - \theta_{v_1|\mathbf{u}_j}^*}{1 - \theta_{v_1|\mathbf{u}_j}} \right|$$

In Chan (2005) the error was corrected by stating that the above expression is an approximation  $\tilde{D}$  of the true distance  $D_p(\Theta_{V|\mathbf{U}}, \Theta_{V|\mathbf{U}}^*)$ .

If we extend Proposition 5 by maximising and minimising over all parent configurations  $\mathbf{u}_j$  as well, we get the exact expression for the distance between two CPTs:  $D_\gamma(\Theta_{V|\mathbf{U}}, \Theta_{V|\mathbf{U}}^*) =$

$$\begin{aligned} & \ln \max_{\mathbf{u}_j} \left\{ \frac{\theta_{v_1|\mathbf{u}_j}^*}{\theta_{v_1|\mathbf{u}_j}}, \frac{1 - \theta_{v_1|\mathbf{u}_j}^*}{\min_{1 \leq k \leq t} \gamma_{v_k|\mathbf{u}_j}^{-1} \cdot \theta_{v_k|\mathbf{u}_j}} \right\} \\ & - \ln \min_{\mathbf{u}_j} \left\{ \frac{\theta_{v_1|\mathbf{u}_j}^*}{\theta_{v_1|\mathbf{u}_j}}, \frac{1 - \theta_{v_1|\mathbf{u}_j}^*}{\max_{1 \leq k \leq t} \gamma_{v_k|\mathbf{u}_j}^{-1} \cdot \theta_{v_k|\mathbf{u}_j}} \right\} \end{aligned}$$

which in the case of proportional co-variation reduces to:  $D_p(\Theta_{V|\mathbf{U}}, \Theta_{V|\mathbf{U}}^*) =$

$$\begin{aligned} & \ln \max_{\mathbf{u}_j} \left\{ \frac{\theta_{v_1|\mathbf{u}_j}^*}{\theta_{v_1|\mathbf{u}_j}}, \frac{1 - \theta_{v_1|\mathbf{u}_j}^*}{1 - \theta_{v_1|\mathbf{u}_j}} \right\} \\ & - \ln \min_{\mathbf{u}_j} \left\{ \frac{\theta_{v_1|\mathbf{u}_j}^*}{\theta_{v_1|\mathbf{u}_j}}, \frac{1 - \theta_{v_1|\mathbf{u}_j}^*}{1 - \theta_{v_1|\mathbf{u}_j}} \right\} \end{aligned}$$

The following example illustrates the difference between this expression and the approximation  $\tilde{D}$ .

**Example 5.** Consider a table  $\Theta_{V|\mathbf{U}}$  for binary-valued  $V$  and  $U$ . Let  $\theta_{v_1|u_1} = 0.8$  and  $\theta_{v_1|u_2} = 0.6$ , and suppose these parameters are decreased to  $\theta_{v_1|u_1}^* = 0.6$  and  $\theta_{v_1|u_2}^* = 0.3$ , respectively. Let  $R$  denote the set of all ratios under consideration, i.e.  $R = \{ \frac{0.6}{0.8}, \frac{1-0.6}{1-0.8}, \frac{0.3}{0.6}, \frac{1-0.3}{1-0.6} \}$ . Then  $D_p(\Theta_{V|\mathbf{U}}, \Theta_{V|\mathbf{U}}^*) = \ln \max R - \ln \min R = \ln 2 - \ln 0.5 = 1.386$ .  $\tilde{D}(\Theta_{V|\mathbf{U}}, \Theta_{V|\mathbf{U}}^*)$ , however, equals the maximum of  $|\ln 0.75 - \ln 2|$  and  $|\ln 0.5 - \ln 1.75|$ , which is 1.252.  $\blacksquare$

In the above example the approximated CD-distance is smaller than the true distance. In fact,

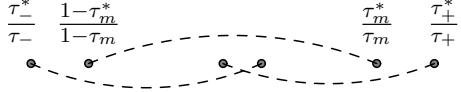


Figure 1: Illustration of distances between (the  $\ln$  of) various parameter ratios. Dashed lines link  $\ln \frac{\tau^*}{\tau}$ -terms to their corresponding  $\ln \frac{1-\tau^*}{1-\tau}$ -terms.

the approximate distance is a lower bound on the true distance.

**Proposition 6.** Let  $\Theta_{V|U}$  and  $\Theta_{V|U}^*$  be as before. Then,

$$D_p(\Theta_{V|U}, \Theta_{V|U}^*) \geq \tilde{D}(\Theta_{V|U}, \Theta_{V|U}^*)$$

*Proof.* Let  $\tau_+^*/\tau_+$  denote the maximum, over all parent configurations  $u_j$ , of  $\theta_{v_1|u_j}^*/\theta_{v_1|u_j}$  and  $(1 - \theta_{v_1|u_j}^*)/(1 - \theta_{v_1|u_j})$ ; likewise, let  $\tau_-^*/\tau_-$  denote the minimum (see Figure 1). Note that  $\tau_+^*/\tau_+ \geq 1$  and  $0 < \tau_-^*/\tau_- \leq 1$ . Then, by definition,

$$D_p(\Theta_{V|U}, \Theta_{V|U}^*) = \ln \frac{\tau_+^*}{\tau_+} - \ln \frac{\tau_-^*}{\tau_-}$$

Now if  $\tau_-$  and  $1 - \tau_+$  actually correspond with the same parameter, then  $D_p(\Theta_{V|U}, \Theta_{V|U}^*) = \tilde{D}(\Theta_{V|U}, \Theta_{V|U}^*)$ , since  $\ln(\tau_+^*/\tau_+) - \ln((1 - \tau_+^*)/(1 - \tau_+))$  equals the largest possible distance.

Otherwise, suppose that  $\tilde{D} = \ln(\tau_m^*/\tau_m) - \ln((1 - \tau_m^*)/(1 - \tau_m))$ , for some  $\tau_m$  such that  $\tau_+^*/\tau_+ \geq \tau_m^*/\tau_m \geq \tau_-^*/\tau_-$ . Then from Figure 1 it is obvious that  $\tilde{D}(\Theta_{V|U}, \Theta_{V|U}^*) \leq D_p(\Theta_{V|U}, \Theta_{V|U}^*)$ .  $\square$

## 6 Conclusions

We have generalised both sensitivity functions and CD-distance to cope with arbitrary co-variation schemes. We showed that the sensitivity function remains a rational function as long as a valid and linear co-variation scheme is used. In addition, we discussed the suitability of various algorithms for computing the sensitivity functions under different co-variation schemes. Finally, we proved a lower bound on CD-distance for single CPTs.

In Section 4.2 we provided some preliminary results on comparing sensitivity functions under various co-variation schemes. We plan to study the differences in more detail in the near future. The

increased complexity of the formula for the CD-distance for single CPT changes, forestalls straightforward generalisation of the known optimality of the CD-distance. If proportional co-variation turns out not to be necessarily optimal for whole CPTs, then alternative co-variation schemes will become more interesting to consider in e.g. the context of parameter tuning. We suspect that sensible co-variation schemes will be domain-dependent, preserving e.g. known thresholds or relationships between parameters.

## References

- E. Castillo, J.M. Gutiérrez, A.S. Hadi (1997). Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 27, pp. 412 – 423.
- H. Chan (2005). *Sensitivity Analysis of Probabilistic Graphical Models*, Ph.D. thesis, University of California, Los Angeles.
- H. Chan, A. Darwiche (2002). When do numbers really matter? *Journal of Artificial Intelligence Research*, 17, pp. 265 – 287.
- 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.
- H. Chan, A. Darwiche (2005). A distance measure for bounding probabilistic belief change. *International Journal of Approximate Reasoning*, 38, pp. 149 – 174.
- 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.
- L.C. van der Gaag, S. Renooij, V.M.H. Coupé (2007). Sensitivity analysis of probabilistic networks. *Advances in Probabilistic Graphical Models*, 213, pp. 103 – 124.
- F.V. Jensen, T.D. Nielsen (2007). *Bayesian Networks and Decision Graphs*, Springer-Verlag.
- 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.

# Tractable Inference in Hybrid Bayesian Networks with Deterministic Conditionals using Re-approximations

Rafael Rumí, Antonio Salmerón

Department of Statistics and Applied Mathematics

University of Almería, Spain

{rrumi,antonio.salmeron}@ual.es

Prakash P. Shenoy

School of Business

University of Kansas, USA

pshenoy@ku.edu

## Abstract

In this paper we study the problem of inference in hybrid Bayesian networks containing deterministic conditionals. The difficulties in handling deterministic conditionals for continuous variables can make inference intractable even for small networks. We describe the use of re-approximations to reduce the complexity of the potentials that arise in the intermediate steps of the inference process. We show how the idea behind re-approximations can be applied to the frameworks of mixtures of polynomials and mixtures of truncated exponentials. Finally, we illustrate our approach by solving a small stochastic PERT network modeled as an hybrid Bayesian network.

## 1 Introduction

Hybrid Bayesian networks are Bayesian networks (BNs) that include a mix of discrete and continuous random variables. The first proposal of an exact algorithm for hybrid Bayesian networks was developed for the case in which the conditional distributions of all continuous variables in the network are conditional linear Gaussian (CLG) (Lauritzen, 1992; Lauritzen and Jensen, 2001), and that discrete variables do not have continuous parents. This implies that the marginal distribution of each continuous variable is a *mixture of Gaussians* (MoG). Some limitations of the MoG model are the CLG assumptions for the conditionals of continuous variables, and the assumption that discrete variables do not have continuous parents.

A more general technique, based on the use of *mixtures of truncated exponentials* (MTEs), was proposed in (Moral et al., 2001). The MTE model does not impose any topological restriction on its corresponding networks, and is com-

patible with any efficient algorithm for exact inference that requires only the combination and marginalization operations, such as the Shenoy-Shafer (Shenoy and Shafer, 1990; Shenoy and West, 2011a) and variable elimination methods (Zhang and Poole, 1996). Furthermore, MTEs have shown a remarkable ability for fitting many commonly used univariate probability density functions (PDFs) (Cobb et al., 2006; Langseth et al., 2010).

A more recent proposal for dealing with hybrid Bayesian networks is based on the use of *mixtures of polynomials* (MOPs) (Shenoy and West, 2011b). Like MTEs, MOPs have high expressive power, but the latter are superior in dealing with deterministic conditionals for continuous variables (Shenoy and West, 2011b; Shenoy and West, 2011a). A conditional distribution for a variable is said to be *deterministic* if its variances are all zeroes for each state of its parents. Also, a MOP approximation of a PDF can be easily found using Lagrange interpolating polynomial with Chebyshev points (Shenoy,

2012). MTEs and MOPs can be seen as special cases of the recently proposed mixtures of truncated basis functions (MoTBFs) (Langseth et al., 2012).

In this paper we discuss the use of a *reapproximation* strategy for making inference tractable. The idea is based on simplifying the potentials that arise in the intermediate steps of the inference process. This simplification is achieved by reducing the number of pieces of MOP/MTE potentials, and also by reducing the degree of MOPs and the number of exponential terms of MTEs. We compare the performance of MOPs and MTEs in this context, through an example arising from a stochastic PERT network (Cinicioglu and Shenoy, 2009).

## 2 MTEs and MOPs

In this section we formally define the MTE and MOP models, which will be used throughout the paper. We will use uppercase letters to denote random variables, and boldfaced uppercase letters to denote random vectors, e.g.  $\mathbf{X} = \{X_1, \dots, X_n\}$ , and its domain will be written as  $\Omega_{\mathbf{X}}$ . By lowercase letters  $x$  (or  $\mathbf{x}$ ) we denote some element of  $\Omega_X$  (or  $\Omega_{\mathbf{X}}$ ). The MTE model (Moral et al., 2001) is defined as follows.

**Definition 1.** Let  $\mathbf{X}$  be a mixed  $n$ -dimensional random vector. Let  $\mathbf{Y} = (Y_1, \dots, Y_d)^T$  and  $\mathbf{Z} = (Z_1, \dots, Z_c)^T$  be the discrete and continuous parts of  $\mathbf{X}$ , respectively, with  $c + d = n$ . We say that a function  $f : \Omega_{\mathbf{X}} \mapsto \mathbb{R}_0^+$  is a *mixture of truncated exponentials (MTE) potential* if for each fixed value  $\mathbf{y} \in \Omega_{\mathbf{Y}}$  of the discrete variables  $\mathbf{Y}$ , the potential over the continuous variables  $\mathbf{Z}$  is defined as:

$$f(\mathbf{z}) = a_0 + \sum_{i=1}^m a_i \exp \left\{ \mathbf{b}_i^T \mathbf{z} \right\}, \quad (1)$$

for all  $\mathbf{z} \in \Omega_{\mathbf{Z}}$ , where  $a_i \in \mathbb{R}$  and  $\mathbf{b}_i \in \mathbb{R}^c$ ,  $i = 1, \dots, m$ . We also say that  $f$  is an MTE potential if there is a partition  $D_1, \dots, D_k$  of  $\Omega_{\mathbf{Z}}$  into hypercubes and in each one of them,  $f$  is defined as in Eq. (1). In this case, we say  $f$  is a  $k$ -piece,  $m$ -term MTE potential.

Mixtures of polynomials (MOPs) were proposed as modeling tools for hybrid Bayesian net-

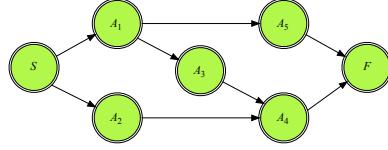


Figure 1: A PERT network with five activities.

works in (Shenoy and West, 2011b). The original definition is similar to MTEs in the sense that they are piecewise functions defined on hypercubes (a more general definition is given in (Shenoy, 2012), where the hypercube condition is relaxed, however it is not used in this paper). The details are as follows.

**Definition 2.** Let  $\mathbf{X}$ ,  $\mathbf{Y}$  and  $\mathbf{Z}$  be as in Def. 1. We say that a function  $f : \Omega_{\mathbf{X}} \mapsto \mathbb{R}_0^+$  is a *mixture of polynomials (MOP) potential* if for each fixed value  $\mathbf{y} \in \Omega_{\mathbf{Y}}$  of the discrete variables  $\mathbf{Y}$ , the potential over the continuous variables  $\mathbf{Z}$  is defined as:

$$f(\mathbf{z}) = P(\mathbf{z}), \quad (2)$$

for all  $\mathbf{z} \in \Omega_{\mathbf{Z}}$ , where  $P(\mathbf{z})$  is a multivariate polynomial in variables  $\mathbf{Z} = (Z_1, \dots, Z_c)^T$ . We also say that  $f$  is a MOP potential if there is a partition  $D_1, \dots, D_k$  of  $\Omega_{\mathbf{Z}}$  into hypercubes and in each one of them,  $f$  is defined as in Eq. (2).

## 3 A PERT Hybrid Bayesian Network

PERT stands for *Program Evaluation and Review Technique*, and is one of the commonly used project management techniques (Malcolm et al., 1959). PERT networks are directed acyclic networks where the nodes represent duration of activities and the arcs represent precedence constraints in the sense that before we can start any activity, all the parent activities have to be completed. The term *stochastic* refers to the fact that the duration of activities are modeled as continuous random variables.

Fig. 1 shows a PERT network with 5 activities  $A_1, \dots, A_5$ . Nodes  $S$  and  $F$  represent the start and finish times of the project. The links among activities mean that an activity cannot be started until after all its predecessors have been completed. Assume we are informed that

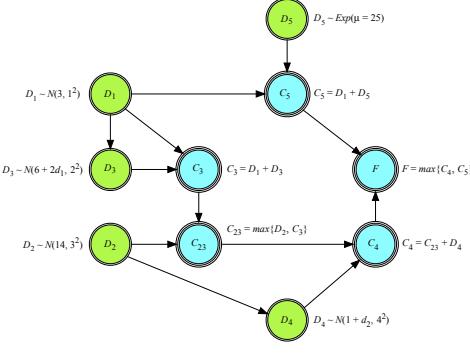


Figure 2: A Bayesian network representing the PERT network in Fig. 1.

the durations of  $A_1$  and  $A_3$  are positively correlated, and the same is true with  $A_2$  and  $A_4$ . Then, this PERT network can be transformed into a Bayesian network as described below.

Let  $D_i$  and  $C_i$  denote the duration and the completion time of the activity  $i$ , respectively. The activity nodes in the PERT network are replaced with activity completion times in the BN. Next, activity durations are added with a link from  $D_i$  to  $C_i$ , so that each activity will be represented by two nodes, its duration  $D_i$  and its completion time  $C_i$ . Notice that the completion times of the activities with no predecessors will be the same as their durations.

We assume that the project start time is zero and each activity is started as soon as all the preceding activities are completed. Thus,  $F$  represents the completion time of the project. The resulting PERT BN is given in Fig. 2.

Notice that the conditionals for the variables  $C_3, C_{23}, C_4, C_5$  and  $F$  are deterministic. On the other hand, variables  $D_1, \dots, D_5$  are continuous random variables, and their corresponding conditional distributions are depicted next to their corresponding nodes in Fig. 2. The parameters  $\mu$  (mean) and  $\sigma^2$  (variance) of the normal distribution are in units of *days* and *days*<sup>2</sup>, respectively. The parameter  $\mu$  (mean) of the exponential distribution is in units of *days*.

Before solving the network, we must deal with the max function in Fig. 2. We convert this max deterministic function to a linear function as proposed in (Shenoy and West, 2011a), transforming the Bayesian network in Fig. 2 into a

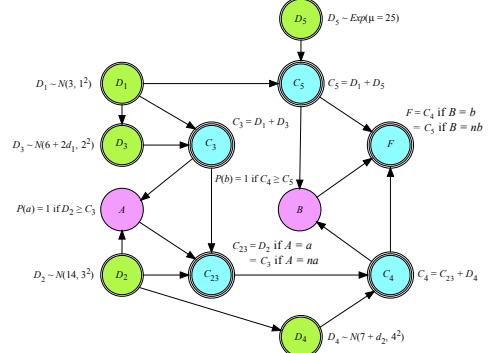


Figure 3: A hybrid Bayesian network representing the PERT network in Fig. 1. Notice that variables  $A$  and  $B$  are discrete.

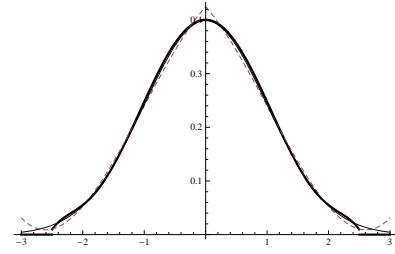


Figure 4: Comparison of the Normal PDF (black), the MTE approximation (thick-black) and the MOP approximation (dashed-black).

hybrid Bayesian network, depicted in Fig. 3.

### 3.1 Solving the Network

The problem of inference in hybrid BNs with deterministic conditionals has been studied in (Shenoy and West, 2011a) and a solution to this PERT network was found in (Shenoy et al., 2011), using the MTEs and MOPs paradigms. When using MOPs, we use a 2-piece 3-degree MOP approximation to the PDF of  $N(0, 1)$  (Shenoy, 2012). In the case of MTEs, we use a 1-piece 7-terms MTE approximation to the PDF of  $N(0, 1)$  (Langseth et al., 2010). The quality of these approximations can be seen in Fig. 4. Using these approximations, we obtain the corresponding approximations to any Gaussian distribution with parameters  $\sigma^2$  and  $\mu$ .

To define the conditional distributions, we followed the mixed tree approach (Moral et al., 2003) for both paradigms. For example, let  $g_2(d_1, d_3)$  be the conditional distribution of

$D_3 \mid d_1$  (the distribution function associated to  $D_3$  in Fig. 3). To define an approximation of  $g_2(d_1, d_3)$  we partition the domain of  $D_1$  into several pieces and assume that  $d_1$  is a constant in each piece equal to the mid-point of the piece:

$$g_{2c}(d_1, d_3) = \begin{cases} f\left(\frac{d_3-8}{2}\right)/2 & \text{if } 0 \leq d_1 < 2, \\ f\left(\frac{d_3-12}{2}\right)/2 & \text{if } 2 \leq d_1 < 4, \\ f\left(\frac{d_3-16}{2}\right)/2 & \text{if } 4 \leq d_1 \leq 6. \end{cases} \quad (3)$$

The number of pieces in which to split the domain (see Eq. (3) above) is a key point to control the computational complexity of the problem. In (Shenoy et al., 2011), the maximum number of pieces in a mixed tree approach was two, which leads to inaccurate solutions. Because we were using a re-approximation strategy (described in Section 4), the number of pieces in the mixed tree approach were increased to three, with the corresponding gain in accuracy.

The goal is to compute the marginal density for  $F$ . To that end, we choose a sequence of the remaining variables in the network,  $D_5 D_3 D_1 D_4 D_2 C_{23} C_3 C_4 C_5$ , and marginalize the variables following this sequence, to compute the marginal density for  $F$ .

Without the re-approximation procedures described in this paper, no solution was obtained to this problem, due to the growth in complexity during the combination and marginalization operations.

## 4 Re-approximation of MOPs/MTEs

In the process of solving the network, we may get pieces for which the density defines a very low or even null probability value. Also, it may happen that the density in several contiguous pieces can be represented as a density in a single larger piece without loss in accuracy. Finally, a third source of complexity may arise from the existence of too many exponential terms (in MTEs) with low contribution to the actual density or a higher degree than necessary (in MOPs). We will describe two methods for re-approximating MOPs and MTEs, consisting of dropping pieces and re-estimating the parameters.

### 4.1 Re-approximation by Dropping Pieces

A simple method of re-approximation is to drop pieces that are defined on lower-dimensional regions. Since there are no probabilities associated with such pieces, dropping them causes no loss of accuracy. In some cases, there may also be pieces with very low probability associated with them. In this case, we can also drop them, as long as the total associated probability is below some threshold (e.g., 0.05) so that the loss of accuracy remains limited. It is necessary to re-normalize the potentials after dropping pieces with positive probabilities.

In the solution of the PERT problem described in Sec. 3, after marginalizing  $D_1$ , we obtain a MOP denoted by  $f_4(c_3, c_5)$  representing the joint PDF of  $C_3$  and  $C_5$ , with 23 pieces, 9 of which defined on 1-dimensional regions. Thus, we can safely drop these pieces resulting in a 14-piece MOP. Further examination of the remaining 14 pieces reveals that 6 of them have very small probabilities (0.000017 to 0.026) associated with them, amounting to a total of  $\approx 0.044$ . After dropping these 6 pieces and re-normalizing the potential, we obtain an 8-piece MOP  $f_{4r}(c_3, c_5)$  that can be used in place of  $f_4(c_3, c_5)$ . Fig. 5 shows plots of  $f_4$  and  $f_{4r}$ .

When solving the network using MTEs, we also get the same potential  $f_4(c_3, c_5)$  with 11-pieces. Of these, 4 are singleton pieces, which can be dropped without loss of accuracy. Three of the remaining pieces have very low probabilities (0.00005 to 0.019) with a total associated probability of 0.026. After dropping these three pieces and re-normalizing the density, we obtain a 4-piece MTE  $f_{4s}(c_3, c_5)$  that can be used in place of  $f_4(c_3, c_5)$ . Fig. 6 shows  $f_4$  and  $f_{4s}$ .

### 4.2 Re-approximation of MOPs using LIP with Chebyshev Points

Another method for re-approximating MOPs is to use Lagrange interpolating polynomials (LIPs) with Chebyshev points (Shenoy, 2012). To illustrate this, consider  $g_6(\cdot)$ , which is defined on non-singleton intervals  $(0, 6]$ ,  $(6, 9]$ ,  $(9, 12]$ ,  $(12, 15]$ ,  $(15, 18]$ ,  $(18, 21]$ ,

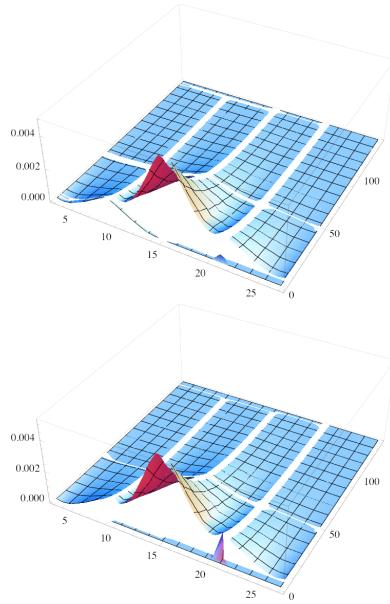


Figure 5: MOPs corresponding to  $f_4$  (top) and  $f_{4r}$  (bottom).

(21, 24), (24, 30). We will describe how to re-approximate  $g_6(\cdot)$  using just 5 pieces, on intervals  $[0, 6)$ ,  $[6, 12)$ ,  $[12, 18)$ ,  $[18, 24)$ ,  $[24, 30]$ . Currently, we do not have a theory for the choice of pieces, so we follow a heuristic strategy consisting of merging pieces with the restriction of keeping the sizes of the pieces approximately equal.

For each interval, we compute  $n$  Chebyshev points with  $n$  representing a tradeoff between complexity and accuracy of the resulting polynomial. A good starting choice is  $n = 4$ . We compute the 3-degree LIP polynomial that passes through the 4 points. We have to verify that the LIP polynomial is non-negative on the interval. If not, we increase  $n$ . If the function being approximated is non-negative over the interval, there are guarantees to find an interpolating polynomial that is non-negative for some  $n$ . This is because when we increase the number of Chebyshev points by 1, the maximum error between the LIP polynomial and the polynomial being approximated is reduced by a factor of 2. If the smallest  $n$  that results in a polynomial that is non-negative is too high, we reduce the width of the interval (i.e., use more pieces) and

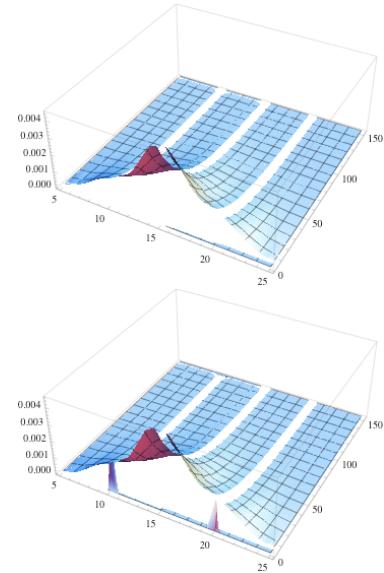


Figure 6: MTEs corresponding to  $f_4$  (top) and  $f_{4s}$  (bottom).

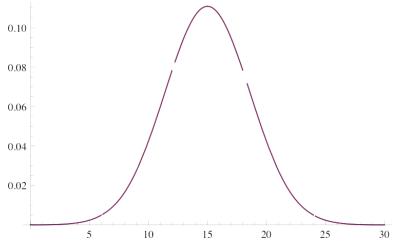


Figure 7: A graph of  $g_{6r}(\cdot)$  overlaid on the graph of  $g_6(\cdot)$  for MOPs.

re-start.

Using  $n = 5$  for all five pieces results in a non negative MOP for  $g_6(\cdot)$ . Next we normalize the resulting 5-piece, 4-degree MOP so that the total area under the MOP is 1. Let  $g_{6r}(\cdot)$  denote the 5-piece, 4-degree MOP found using the above procedure. Fig. 7 shows  $g_{6r}(\cdot)$  overlaid on  $g_6(\cdot)$ .

The LIP method applies for two or higher dimensional functions. There also exists Chebyshev points theory for two-dimensional regions (Xu, 1996). For regions in 3 or higher dimensions, we can use some extensions of the one-dimensional Chebyshev point theory. One problem with using LIPs with Chebyshev points in two or higher dimensions is non-negativity. It may be necessary to increase the degree of the

MOP potential to a very high number making computations numerically unstable. In such cases, we can resort to the idea behind mixed-tree approximations, and use a one-dimensional LIP method with Chebyshev points to approximate a two or higher dimensional function.

To illustrate this, consider  $f_7(c_3, c_4)$  representing the conditional PDF of  $C_4$  given  $c_3$  in the PERT hybrid BN discussed in Section 3.  $f_7$  is computed as a 49-piece, 7-degree MOP on the domain  $(3 \leq c_3 \leq 27) \times (1.125 \leq c_5 \leq 137.5)$ . We will approximate this potential by a 8-piece, 7-degree MOP  $f_{7r}$  as follows. Each of the 49 pieces of  $f_7$  are re-defined on a 4-way partition of  $(3 \leq c_3 \leq 27)$ :  $[3, 11]$ ,  $[11, 17]$ ,  $[17, 23]$ , and  $[23, 27]$ . Consider the region  $3 \leq c_3 < 11$ . We approximate  $f_7(7, c_4)$  (a 1-dimensional function,  $c_3 = 7$  is the mid-point of the interval) using LIP with Chebyshev points as discussed earlier by a 2-piece, 7-degree MOP and use this approximation for the region  $(3 \leq c_3 < 11) \times (11 \leq c_5 \leq 59)$ . By doing this for all four pieces of the partition of the domain of  $C_3$ , we obtain a 8-piece, 7-degree MOP approximation  $f_{7r}$  of the two-dimensional MOP  $f_7$ . Fig. 8 displays  $f_7$  and  $f_{7r}$ .

### 4.3 Re-approximation of MTEs using Least Squares

In this section, we show how to re-approximate MTEs by reducing the number of pieces and exponential terms. The idea behind this method is similar to the one used for MOPs, but using a different mathematical tool. In this method, to re-approximate a function  $f$ , we use the command *FindFit* implemented in *Mathematica*®, which performs a numerical least-squares fit. A key point in this procedure is selecting appropriate starting values for the set of parameters to estimate. Since the MTE approximation to the Gaussian distribution is accurate, we will use the parameter estimates of this approximation as starting values in the *FindFit* method.

The steps to approximate  $f$  are as follows. 1: Divide the domain of  $f$  in a number of pieces lower than the original one. 2: For each new piece, compute the mean and standard deviation using the original density, and fit an

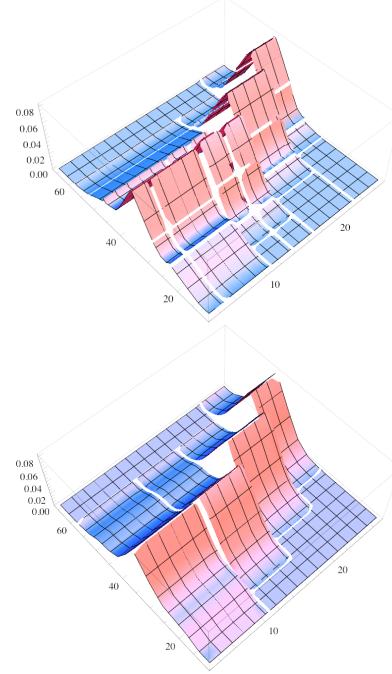


Figure 8: 3-D plots of  $f_7$  (top) and  $f_{7r}$  (bottom) for MOPs.

MTE to the corresponding Gaussian density with those parameters. 3: Select a sample of equally distributed values of  $X$  in the current piece (25 values were selected in this example). 4: Build a 2-dimensional sample  $(x_i, f(x_i))$ . 5: Define an MTE potential in the corresponding piece, fixing the number of terms. 6: Use  $(x_i, f(x_i))$  and the starting values to compute the estimates of the parameters of the MTE potential. 7: Re-normalize each piece so that the probability of the re-approximated potential is the same as the original for each piece.

Notice that this procedure is flexible. For example, if the function being re-approximate is not too complicated, we can set a different number of terms in different regions, as we actually do in this PERT network in some re-approximations. We use 6-terms for difficult approximations, usually central regions, and 2-terms for simpler regions, usually tails. Like MOPs, there is not yet a straightforward way to divide the domain, so we followed the strategy of dividing it if the shape of the function differs from Gaussian shape. This is also related to a

successful selection of the starting points, since it requires that the density in the piece being re-approximated somehow resembles a Gaussian shape, which is not always the case.

In the case of re-approximating 2 or higher dimensional potentials, we use the same procedure as MOPs, i.e., using mixed trees and fixing one dimension whilst re-approximating the function for the remaining dimensions. In the case of MTEs this is not difficult, since this is the approach selected to define the conditional distributions (parent variables can only affect the partition of the domain, not the expression of the density itself).

Such a procedure was carried out to re-approximate  $f_7(c_3, c_4)$ , a 54-piece potential (many of them singleton points) and between 7 and 65 terms (depending on the hypercube). The result is an approximate potential with 7 pieces, and with 6 terms for 3 of the pieces and 2 terms for the remaining 4 pieces. In Fig. 9 we can see the original MTE potential and the corresponding re-approximated potential.

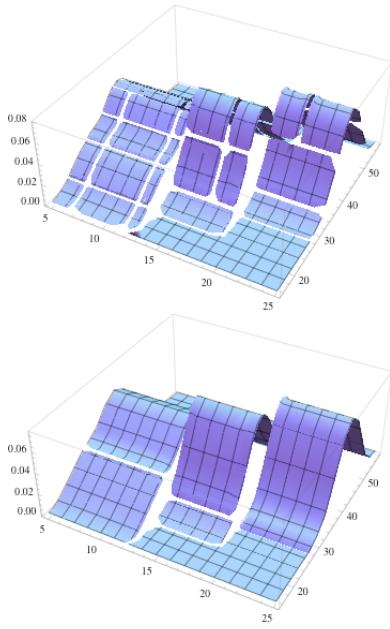


Figure 9: 3-D plots of  $f_7$  (top) and  $f_{7r}$  approximated (bottom) for MTEs.

Notice that that the re-approximation strategy can lead to very accurate solutions, in which the complexity reduction is minimal, or to very

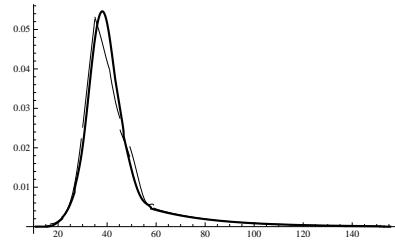


Figure 10: Plot of MOP (black) and MTE (thick-black) density for  $F$ .

smooth approximations, but with a big decrease in the complexity. It depends on the number of pieces selected and the number of exponential terms included (or degree of the polynomial in the MOP case). Finding a tradeoff between these two extremes is still an open question.

Implementing re-approximations in the elimination procedure to solve the PERT network lead to two different solutions for  $F$ . Using MOPs, we obtained a solution in 468.54 seconds, in which  $E(F) = 43.44$  and  $Var(F) = 246.87$ . Without re-approximations we were not able to obtain a solution. Using MTEs, we obtained a solution in 481.90 seconds, in which  $E(F) = 43.78$  and  $Var(F) = 260.74$ . Solving the same problem using MTEs without re-approximations took 1,600 seconds (Shenoy et al., 2011). Fig. 10 shows graphs of the MTE and the MOP densities for  $F$ .

## 5 Summary and Conclusions

The main contribution of this paper is a re-approximation strategy for making computations with MOPs and MTEs tractable in hybrid BN with deterministic conditionals. The significance of this contribution is that we can now solve some hybrid BNs that we could not solve before. The re-approximations results in a reduction in the sizes of the potentials involved in the solution process, and also results in a reduction in the complexity of the potentials (in terms of degree of the polynomials and number of exponential terms). The mathematical tools used for re-approximation are LIP and numerical least squares, and both of them can be implemented using software such

as *Mathematica*®, which broadens the potential users of hybrid BNs in communities beyond computer science and statistics.

The re-approximation strategy can serve as a basis for designing algorithms that scale up to solve large networks. Our future research will further investigate this issue. Another line of future work is to extend the re-approximation strategy by using the approximation method presented in (Langseth et al., 2012) in the context of MoTBFS.

## Acknowledgments

This research has been funded in part by the Spanish Ministry of Economy and Competitiveness through projects TIN2010-20900-C04-01,02, by Junta de Andalucía through project P11-TIC-7821, by the European Regional Development Funds (FEDER), and by a sabbatical from the University of Kansas.

## References

- E. N. Cinicioglu and P. P. Shenoy. 2009. Using mixtures of truncated exponentials for solving stochastic PERT networks. In J. Vejnarová and T. Kroupa, editors, *Proceedings of the Eighth Workshop on Uncertainty Processing*, pages 269–283. Univ. of Economics, Prague.
- B. R. Cobb, P. P. Shenoy, and R. Rumí. 2006. Approximating probability density functions with mixtures of truncated exponentials. *Statistics and Computing*, 16(3):293–308.
- H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. 2010. Parameter estimation and model selection for mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 51(5):485–498.
- H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. 2012. Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53(2):212–227.
- S. L. Lauritzen and F. Jensen. 2001. Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11(2):191–203.
- S. L. Lauritzen. 1992. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108.
- D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar. 1959. Application of a technique for research and development program evaluation. *Operations Research*, 7(5):646–669.
- S. Moral, R. Rumí, and A. Salmerón. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In S. Benferhat and P. Besnard, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Lecture Notes in Artificial Intelligence 2143, pages 135–143. Springer, Berlin.
- S. Moral, R. Rumí, and A. Salmerón. 2003. Approximating conditional MTE distributions by means of mixed trees. In T. D. Nielsen and N. L. Zhang, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Lecture Notes in Artificial Intelligence 2711, pages 173–183. Springer, Berlin.
- P. P. Shenoy and G. Shafer. 1990. Axioms for probability and belief function propagation. In R. D. Shachter, T. S. Levitt, J. F. Lemmer, and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence 4*, pages 169–198. North Holland, Amsterdam.
- P. P. Shenoy and J. C. West. 2011a. Extended Shenoy-Shafer architecture for inference in hybrid Bayesian networks with deterministic conditionals. *International Journal of Approximate Reasoning*, 52(6):805–818.
- P. P. Shenoy and J. C. West. 2011b. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52(5):641–657.
- P. P. Shenoy, R. Rumí, and A. Salmerón. 2011. Some practical issues in inference in hybrid Bayesian networks with deterministic conditionals. In S. Ventura et al., editor, *Proceeding of the 11th International Conference on Intelligent Systems Design and Applications*, pages 605–610. IEEE Research, Piscataway, NJ.
- P. P. Shenoy. 2012. Two issues in using mixtures of polynomials for inference in hybrid Bayesian networks. *International Journal of Approximate Reasoning*, 53(5):847–866.
- Y. Xu. 1996. Lagrange interpolation on Chebyshev points of two variables. *Journal of Approximation Theory*, 87(2):220–238.
- N. L. Zhang and D. Poole. 1996. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328.

# Mixtures of Bagged Markov Tree Ensembles

François Schnitzler, Pierre Geurts, Louis Wehenkel

Université de Liège, Belgium

fschnitzler@ulg.ac.be, P.Geurts@ulg.ac.be, L.Wehenkel@ulg.ac.be

## Abstract

Markov trees, a probabilistic graphical model for density estimation, can be expanded in the form of a weighted average of Markov Trees. Learning these mixtures or ensembles from observations can be performed to reduce the bias or the variance of the estimated model. We propose a new combination of both, where the upper level seeks to reduce bias while the lower level seeks to reduce variance. This algorithm is evaluated empirically on datasets generated from a mixture of Markov trees and from other synthetic densities.

## 1 Introduction

Estimating a multivariate probability density over a set  $\mathcal{X}$  of  $p$  variables  $X_1, \dots, X_p$ , based on a sample of joint observations of these variables, is an important challenge to deal with the uncertainty inherent to many fields, such as bioinformatics or fault diagnostics. A well-known framework to encode multivariate densities is the class of Bayesian Networks. Such a probabilistic graphical model defines a factorization of the joint probability distribution :

$$\mathbb{P}_{\mathcal{G}}(\mathcal{X}) = \prod_{i=1}^p \mathbb{P}_{\mathcal{G}}(X_i | Pa_{\mathcal{G}}(X_i)) , \quad (1)$$

where  $\mathcal{G}$  is the graphical structure of the Bayesian network (a directed acyclic graph whose vertices are in bijection with  $\mathcal{X}$ ) and  $Pa_{\mathcal{G}}(X_i)$  denotes the subset of variables parent of  $X_i$  in  $\mathcal{G}$ .  $\mathcal{G}$  encodes conditional independence relationships implied by this factorization (Pearl, 1988; Koller and Friedman, 2009).

A structure  $\mathcal{G}$  of a Bayesian network can be learned from a sample  $\mathbf{D}$  of  $N$  observations generated from the target density. This is typically done by optimizing a decomposable score (maximum likelihood, BIC, AIC...) over the space of candidate structures, or by matching a set of conditional independences inferred from  $\mathbf{D}$ . Learning the parameters of the conditional densities  $\mathbb{P}_{\mathcal{G}}(X_i | Pa_{\mathcal{G}}(X_i))$  essentially amounts

to counting in  $\mathbf{D}$  the frequencies of joint configurations of the variables  $\{X_i\} \cup Pa_{\mathcal{G}}(X_i)$ .

A Bayesian network model can be exploited to answer probabilistic queries, such as estimating the conditional density of a subset of variables given the values of another subset of variables. This process is also called inference.

The complexity of learning (Chickering et al., 1994) and inference (Cooper, 1990; Kwisthout et al., 2010) for the class of Bayesian networks is however NP-hard, and in practice not possible beyond a few thousand variables (Auvray and Wehenkel, 2008; Elidan and Gould, 2008). With the rapid expansion in data acquisition technology, many applications exceed this limit. One strategy to tackle this problem is to constrain the structures considered, e.g. (Bach and Jordan, 2001; Elidan and Gould, 2008).

Markov trees are such a restricted subclass of Bayesian networks, where the graphical structure underlying any model is connected and  $|Pa_{\mathcal{G}}(X_i)| \leq 1$  (Pearl, 1988). This subclass is particularly interesting: constructing the structure maximizing the likelihood of the sample  $\mathbf{D}$  has a quadratic complexity in the number of variables while inference has linear complexity.

Mixtures of Markov trees are a convex combination of a set  $\hat{\mathcal{T}} = \{T_1, \dots, T_m\}$  of  $m$  elementary Markov tree densities. They were introduced in (Meila and Jordan, 2001) to improve on Markov trees while retaining their interesting scaling behavior with respect to the num-

ber of variables. There are two frameworks for building those mixtures. The first one attempts to improve the modeling capacity of Markov trees, which may be insufficient to model a problem correctly due to the structural constraints. Learning the mixture is in that case viewed as a global optimization problem aiming at maximizing the data likelihood (Meila and Jordan, 2001). In the second framework,  $\hat{T}$  can be viewed as a set of different alternatives that cannot be discriminated based on the data set. Considering different possibilities obtained by a randomized algorithm reduces the variance of the Chow-Liu tree (Ammar et al., 2010). Another method to reduce overfitting is regularization, but it requires tuning the regularization strength. It has been applied to Markov trees as well (Tan et al., 2011; Liu et al., 2011).

These two types of mixtures are combined by Kollin and Koivisto (2006), who define a Markov-Chain Monte Carlo over k-clusterings of the observations and construct one term on each cluster by exact Bayesian averaging over the space of Markov trees (Meila and Jaakkola, 2006); and by Kirshner and Smyth (2007), who use a similar approach but allow k to vary and sample one tree from the Bayesian posterior. While both papers report an improvement over the original mixture of Meila and Jordan (2001), these methods are of cubic complexity in the number of variables, thus deteriorating the scalability of Markov trees.

In this work we develop a more scalable (quadratic) alternative to combine these two frameworks, by replacing Markov trees in the first approach (maximum likelihood, ML) by a mixture of trees generated by the second (model averaging). Rather than learning a single Markov tree by the Chow-Liu algorithm for each term of the upper mixture, a randomized procedure is used to replace each term by a lower level mixture. The combination order is thus the opposite of (Kirshner and Smyth, 2007). This opposite order is essentially motivated by computational considerations.

We begin by recalling the Chow-Liu algorithm in Section 2 and presenting mixtures of

---

**Algorithm 1** Chow-Liu algorithm

---

**Input:** data  $\mathbf{D}$   
 $M = [0]_{p \times p}$   
**for**  $i_1 = 1, i_2 = i_1 + 1$  **to**  $p, p$  **do**  
     $M[i_1, i_2] = I_{\mathbf{D}}(X_{i_1}; X_{i_2})$   
**end for**  
**return**  $T = \text{MWST}(M)$

---

Markov trees more formally in Section 3. The proposed approach is described in Section 4, its interest and variance reduction capacity are experimentally studied in Section 5.

## 2 Markov Trees

The Chow-Liu algorithm (1968) is used as a building block in most methods for learning a mixture of Markov trees. It outputs the Markov tree structure maximizing the likelihood of the learning sample  $\mathbf{D}$ , the solution of the following optimization problem :

$$T_{CL}(\mathbf{D}) = \arg \max_T \sum_{(X_{i_1}, X_{i_2}) \in \mathcal{E}(T)} I_{\mathbf{D}}(X_{i_1}; X_{i_2}), \quad (2)$$

where  $\mathcal{E}(T)$  is the set of edges of the tree  $T$ .

The Chow-Liu algorithm (algorithm 1) first estimates the mutual information  $I_{\mathbf{D}}(X_{i_1}; X_{i_2})$  between any pair of variables  $X_{i_1}$  and  $X_{i_2}$ , hence defining the symmetric adjacency matrix  $M$  of a complete graph. In a second step, a maximum-weight spanning tree is computed based on  $M$ , yielding an undirected graph which may be directed by choosing an arbitrary node as root.

## 3 Mixtures of Markov Trees

Working in the class of Markov trees is algorithmically efficient, but can be either too restrictive, i.e. have a large bias, when samples are plentiful, or lead to overfitting, i.e. have a large variance, when only few samples are available.

Mixtures of Markov trees can reduce either shortcoming of Markov trees while preserving their algorithmic scalability in  $p$ . Such a mixture  $\{\hat{T}, w\}$  is composed of a set  $\hat{T} = \{T_1, \dots, T_m\}$  of  $m$  elementary Markov tree densities and a set  $w = \{w_i\}_{i=1}^m$  of weights, yielding

a density in the form of a convex combination:

$$\mathbb{P}_{\hat{T}}(\mathcal{X}) = \sum_{k=1}^m w_k \mathbb{P}_{T_k}(\mathcal{X}) . \quad (3)$$

This is a proper density, assuming  $w_i \in [0, 1]$  and  $\sum_{i=1}^m w_i = 1$ . Those two frameworks are briefly presented below, each with an exemplificative algorithm, also used as building blocks in the combination proposed in Section 4.

### 3.1 Mixture for Reducing the Bias

Constraining a Bayesian network to a Markov tree when the target distribution is more complex than such a tree and for a sufficiently large number of samples will cause some relationships between variables to be missed and the distribution won't be perfectly estimated, an error called the bias. Using a mixture rather than a single Markov tree results in an improved modeling power (Meila and Jordan, 2001) and a higher achievable likelihood.

This optimization can be done e.g. by the EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2008). The weights are considered as the marginal probabilities  $\mu_k = \mathbb{P}(Z = k)$  of  $Z$ , a hidden variable selecting one distribution  $\mathbb{P}_{T_k}(\mathcal{X}) = \mathbb{P}(\mathcal{X}|Z = k)$ :

$$\mathbb{P}_{\hat{T}}(\mathcal{X}) = \sum_{k=1}^m \mathbb{P}(Z = k) \mathbb{P}(\mathcal{X}|Z = k) . \quad (4)$$

This estimate of  $\mathbb{P}(\mathcal{X}, Z)$  is optimized by alternating between estimating a distribution of the hidden variable  $Z$  for each observation  $\mathbf{D}[i]$  and optimizing  $\mathbb{P}(Z = k)$  and  $\mathbb{P}(\mathcal{X}|Z = k)$ . This process is described in algorithm 2, where  $\gamma_k(i)$  can be seen as the probability that  $Z = k$  for observation  $\mathbf{D}[i]$  and according to the current estimate of  $\mathbb{P}(\mathcal{X}, Z)$ . The vectors  $(\gamma_1(i), \dots, \gamma_m(i))_{i=1}^N$  define a soft partition of  $\mathbf{D}$  into  $m$  weighted learning samples  $\mathbf{D}'_k$ , where each  $\mathbf{D}'_k$  is constructed by associating a weight  $\gamma_k(i)$  to each  $\mathbf{D}[i]$ . By comparison, all observations in  $\mathbf{D}$  have a weight of one. In the second step of the iteration, the Chow-Liu algorithm is applied to each  $\mathbf{D}'_k$  to obtain  $T_k$  and  $\mu_k$  is estimated based on the  $\gamma_k(i)$ .

---

**Algorithm 2** MT-EM: ML mixture of Markov trees

---

```

Input: data \mathbf{D} , mixture size m
Initialize $\{\hat{T}, \mu\}$
repeat
 for $k = 1$ to m and $i = 1$ to N do
 $\gamma_k(i) = \frac{\mu_k \mathbb{P}_{T_k}(\mathbf{D}[i])}{\sum_{k=1}^m \mu_k \mathbb{P}_{T_k}(\mathbf{D}[i])}$
 end for
 for $k = 1$ to m do
 $\mathbf{D}'_k = (\mathbf{D}[i], \gamma_k(i))_{i=1}^N$
 $T_k = \text{Chow-Liu}(\mathbf{D}'_k) ; \mu_k = \frac{\sum_{i=1}^N \gamma_k(i)}{N}$
 end for
 until convergence
return mixture of Markov trees $\{\hat{T}, \mu\}$

```

---

### 3.2 Mixture for Reducing the Variance

When the number of available samples is small, two datasets  $\mathbf{D}_1$  and  $\mathbf{D}_2$  will probably result in two different models, and the smaller the sample size  $N$ , the greater the difference. This is called the variance. When this variance is too high, the models are said to be overfitting the data, i.e. they are modeling noise caused by the low number of samples. To reduce this effect, the second framework builds mixtures as an average over different plausible models.

A Bayesian approach where all trees are averaged is possible (Meila and Jaakkola, 2006), but it is of cubic complexity in  $p$  and does not permit inference. Although sampling models from this distribution is possible, we want to retain the quadratic complexity of the Chow-Liu algorithm. This is possible using the “perturb and combine” framework, which was really successful in supervised learning and can also be applied to density estimation. It consists in perturbing a learning algorithm to randomize it (making it suboptimal), and in combining the models output by different calls to this randomized algorithm.

Bagging (Breiman, 1996) is a widely used method of this category that consists in applying a given learning algorithm on  $m$  bootstrap replicates of an original data set  $\mathbf{D}$ , resulting in  $m$  different models. A bootstrap replicate

**Algorithm 3** MT-Bag: bagging Markov trees

---

```

Input: data \mathbf{D} , mixture size m
 $\hat{\mathcal{T}} = \emptyset, \lambda = \{1/m, \dots, 1/m\}$
for $k = 1$ to m do
 $\mathbf{D}' = \text{bootstrap}(\mathbf{D})$
 $\hat{\mathcal{T}} = \hat{\mathcal{T}} \cup \{\text{Chow-Liu}(\mathbf{D}')\}$
end for
return mixture of Markov trees $\{\hat{\mathcal{T}}, \lambda\}$

```

---

is constructed by randomly sampling (with replacement) observations from  $\mathbf{D}$  and copying them in the replicate. Typically, the replicate has the same size as  $\mathbf{D}$ .

Algorithm 3 describes bagging applied to Markov trees. The resulting set  $\hat{\mathcal{T}}$  associated to uniform weights can be viewed as a mixture.

## 4 Two-level Mixtures

In this section we describe our new algorithm to combine the methods presented in the previous section. Combining a bias and a variance reducing framework has also been considered in supervised learning, see e.g. (Breiman, 1999).

The EM algorithm builds a soft partition of size  $m$  of the learning set and one maximum-likelihood tree on each downweighted data set  $\mathbf{D}'_k$ . Therefore each tree is built using a smaller effective number of samples than in the original data set. On the other hand, algorithms of the second category build mixtures that are increasingly better than a single tree when the number of samples shrinks (Schnitzler et al., 2010).

We therefore propose to replace the Chow-Liu step in algorithm 2 by an algorithm of the variance reduction framework to learn each term of the ML mixture. Each algorithm can operate in the configuration it excels in. MT-EM operates on the whole, preferably large, learning set at the upper level, to decrease the bias. MT-Bag works on each smaller subset of observations at the lower level, to reduce the variance. The resulting model is thus a two-level mixture of Markov trees of the following form:

$$\mathbb{P}_{\hat{\mathcal{T}}}(\mathcal{X}) = \sum_{k=1}^{m_1} \mu_k \mathbb{P}_{\hat{\mathcal{T}}_k}(\mathcal{X}) \quad (5)$$

**Algorithm 4** MT-EM-Bag: two-level mixture

---

```

Input: data \mathbf{D} , mixture sizes m_1, m_2
 $\{\hat{\mathcal{T}}, \mu\} = \text{MT-EM}(\mathbf{D}, m_1)$
for $i = 1, k = 1$ to N, m_1 do
 $\gamma_k(i) = \frac{\mu_k \mathbb{P}_{T_k}(\mathbf{D}[i])}{\sum_{k=1}^m \mu_k \mathbb{P}_{T_k}(\mathbf{D}[i])}$ {EM weights}
end for
for $k = 1$ to m_1 do
 $\mathbf{D}'_k = (\mathbf{D}[i], \gamma_k(i))_{i=1}^N$
 $\{\hat{\mathcal{T}}_k, \lambda_k\} = \text{MT-Bag}(\mathbf{D}'_k, m_2)$
end for
return mixture of Markov trees $\{\hat{\mathcal{T}}, \lambda\}$

```

---

$$\mathbb{P}_{\hat{\mathcal{T}}_k}(\mathcal{X}) = \sum_{j=1}^{m_2} \lambda_{k,j} \mathbb{P}_{T_{k,j}}(\mathcal{X}) \quad \forall k \in [1, m_1] , \quad (6)$$

where  $m_1$  and  $m_2$  are respectively the number of terms in the upper level EM optimization of the mixture and in the lower level mixtures  $\hat{\mathcal{T}}_k$  learned on each  $\mathbf{D}'_k$ ;  $T_{k,j}$  is the  $j$ th Markov tree in  $\hat{\mathcal{T}}_k$  and  $\lambda_{k,j}$  denotes its weight.

Using algorithms for reducing the variance rather than the Chow-Liu algorithm during the iterative process of the EM algorithm substantially slows down each step and alters the convergence, resulting in a lower accuracy than MT-EM. Hence, we substitute MT-Bag to Chow-Liu only after a first run until convergence of the basic EM algorithm using single trees and do not modify the  $\gamma_k(i)$ . The resulting method is specified by algorithm 4.

Note that bagging is now performed on a weighted data set. The samples composing the replicates in MT-Bag are thus no longer drawn uniformly, but based on a distribution where the probability of selecting sample  $i$  equals  $\gamma_k(i) / \sum_{i=1}^N \gamma_k(i)$ .  $\sum_{i=1}^N \gamma_k(i)$  (truncated to an integer) observations are drawn (and associated to a weight of 1) to form each replicate.

## 5 Experimental Validation

We consider two strategies for building the second level mixture. In addition to bagging Markov trees (algorithm 4, MT-EM-Bag), we consider locking the first tree in each second-level mixture to the Chow-Liu tree for that term, i.e. a mix of the first two methods (MT-

EM-BagCl). In our implementation, the parameters of a tree are always learned from the full  $\mathbf{D}'_k$  and not from a bootstrap replicate, since it improves accuracy (Schnitzler et al., 2010).

We evaluate the interest of these combinations against the baseline algorithm 2 described in Section 3. The EM algorithm may converge to local maxima, so in order to remove the influence of its random initialization, every comparison between the methods is performed based on mixtures built on similar convergence points of the EM algorithm. In other words, for every run of the EM algorithm, we use its final soft partition of the learning set ( $\gamma_k(i)$ ) to build one two-level mixture based on each algorithm considered. We also use the weights  $\mu_k$  returned by this algorithm to formulate the resulting model.

The evaluation is performed using two different target distribution settings:

1. 1 mixture model of three randomly generated Markov trees defined on 100 variables;
2. 5 synthetic relatively sparse Bayesian networks of 200 variables, generated by randomly sampling for each variable  $X_i$ ,  $k_i$  parents in  $\{X_1, \dots, X_{i-1}\}$ , where  $k_i$  is randomly sampled in  $[0, \min(i-1, 5)]$ ;

The accuracy of any model  $\mathbb{P}_{\hat{T}}$  constructed is quantified using a Monte Carlo estimate of the Kullback-Leibler divergence, with respect to the target distribution  $\mathbb{P}$ :

$$\hat{D}_{KL}(\mathbb{P} \parallel \mathbb{P}_{\hat{T}}) = \frac{1}{N'} \sum_{x \sim \mathbb{P}}^{N'} \log_e \left( \frac{\mathbb{P}(x)}{\mathbb{P}_{\hat{T}}(x)} \right), \quad (7)$$

where  $N'$  the number of observations  $x$  for the estimate equals 10000 in the first case and 50000 in the second. The same samples are used for every evaluation to a given target distribution.

In the first setting, the similarity of any learned mixture to the target mixture was also assessed, using the weight of the maximum weighted bipartite matching between the terms in the estimated mixture and those of the target model. For MT-EM-Bag and MT-EM-BagCl, an original Markov tree  $T$  is compared to a mixture through the average number of common edges between  $T$  and each tree of the mixture.

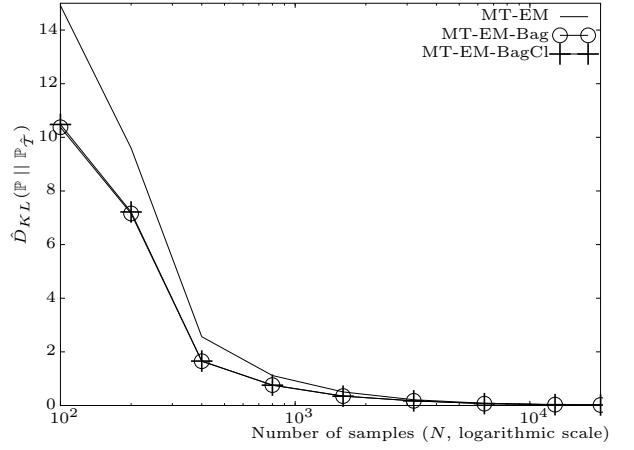


Figure 1: Setting 1: The Average KL divergence with respect to the target mixture of 3 trees converges to zero for all methods as  $N$  increases.

Below  $m_2$  is always 10 for the lower level mixtures. Increasing  $m_2$  further reduces variance without effect on bias, and so is liable to further increase the accuracy of the models.

Due to space restrictions, data sets, supplementary results and graphics (including on more realistic problems) are provided online<sup>1</sup>.

### 5.1 The Target is a Mixture of 3 Trees

A first set of experiments is performed on a target distribution belonging to the class of models considered by the EM learning algorithm, here the set of mixtures of 3 Markov trees.

The number of upper-level terms  $m_1$  is always fixed to 3 (since in this context the size of the target mixture is known a priori); the number  $m_2$  of lower level terms is fixed to 10 to limit execution time. Increasing  $m_2$  may further reduce the variance and improve the accuracy. Learning samples sizes from 100 to 20000 were considered, and several initializations (905 to 50 runs, see Figure 3) were performed for each sample size.

Figure 1 displays the decrease of the KL divergence to zero as  $N$  increases, as expected since the target distribution is a mixture of 3 Markov trees. We observe that the two-level mixtures are better than the baseline, showing the interest of their variance reduction. Actu-

<sup>1</sup>[www.montefiore.ulg.ac.be/~schnitzl/PGM2012/](http://www.montefiore.ulg.ac.be/~schnitzl/PGM2012/)

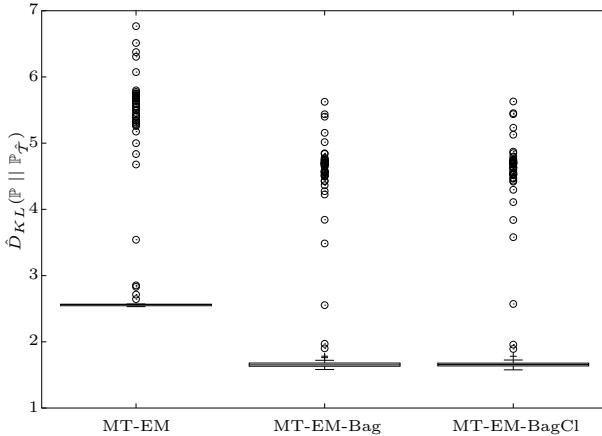


Figure 2: Setting 1: For each method, a box-plot showing the distribution of the KL divergences of the models it produces over 400 runs on a dataset of  $N = 400$  samples.

ally, MT-EM-Bag and MT-EM-BagCl achieve with  $N$  samples roughly the same accuracy as the baseline MT-EM method with  $2N$  samples.

The advantage of the two-level mixtures over MT-EM is significant, as can be seen from the difference between the KL divergence of each method and the KL divergence of the corresponding EM method. This is illustrated in Figure 2, where each box-plot shows the distribution of KL values obtained over 400 runs of MT-EM on a learning set ( $N = 400$ ); similar results are obtained for other sample sizes.

In order to determine which one of the two-level methods is better, Figure 3 displays for each learning sample size the relative number of runs where each method achieves a lower KL divergence than the two others. No definite winner among the two proposed methods can be selected based on those results, but we observe that MT-EM is never better than both (actually, any) of them. However, the interest of using the original learning set to build the first tree of the mixture (MT-EM-BagCl) rather than a bootstrap replicate (MT-EM-Bag) increases with the number of samples.

The edge similarities displayed in Figure 4 reveal an expected increase with the number of samples, but also that the EM algorithm is here the best method. To infer the structure of a mixture of trees, the two-level mixtures are

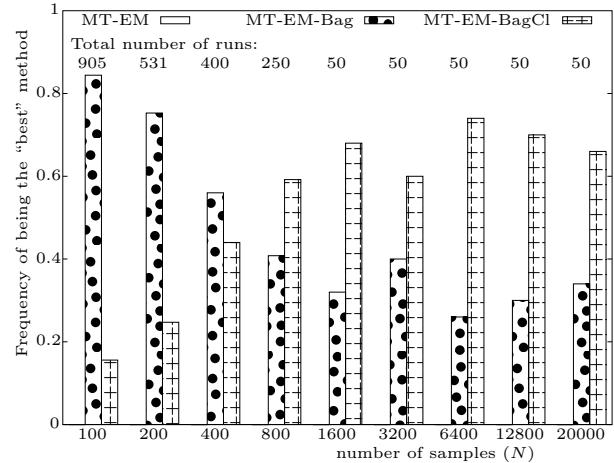


Figure 3: Setting 1: Relative number of runs where each method is the best, displayed by number of samples. MT-EM is always at 0.

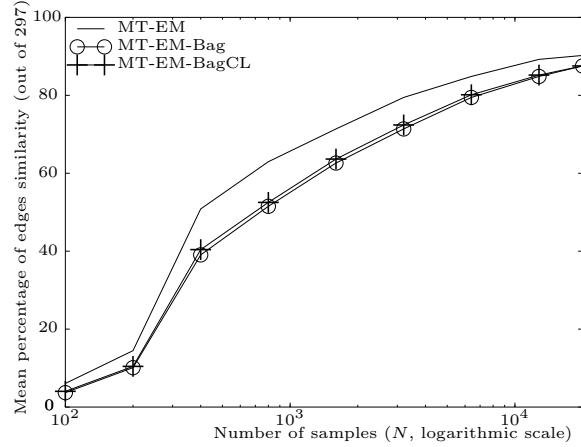


Figure 4: Setting 1: The average edge similarity increases with the learning sample size. MT-EM is better for structure recovery.

thus not as effective. However, this also indicates that their better accuracy is indeed due to the diversity introduced in each term.

## 5.2 The Target is a Bayesian Network

The proposed algorithms were also tested on distributions encoded by a Bayesian network. The optimal number of terms  $m_1$  in the EM method is problem dependent and is e.g. determined by cross-validation. In particular, this value tends to increase with  $N$ , with the Chow-Liu algorithm ( $m_1 = 1$ ) being better than EM ( $m_1 > 1$ ) for small  $N$ . In this section we investigate how varying  $N$  and  $m_1$  impact

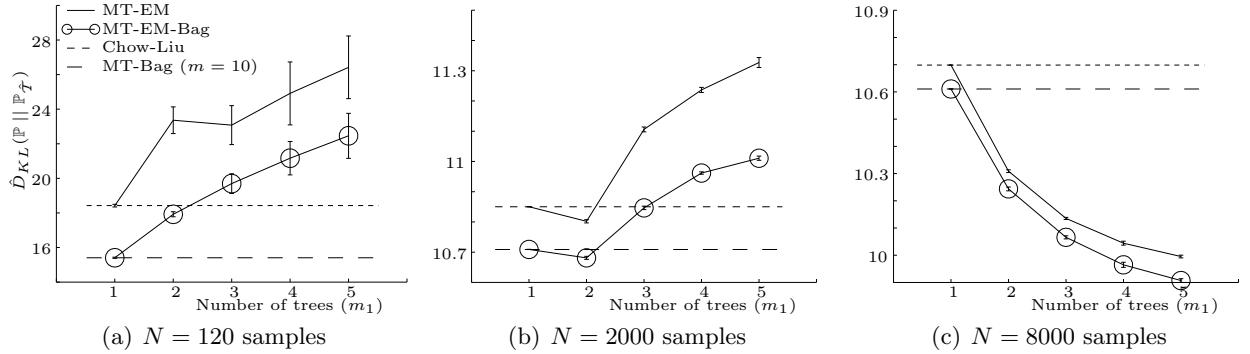


Figure 5: Setting 2: On 1 run times 5 distributions times 6 sets, increasing  $m_1$  ( $m_2 = 10$ ) reduces the bias and increases the variance. While a mixture is advantageous only for  $N$  large enough, MT-EM-Bag is always better than MT-EM.

the difference between the methods and the bias/variance trade-off.

Figure 5 displays for growing values of  $N$  the mean accuracy of MT-EM and MT-EM-Bag as a function of  $m_1$  and with  $m_2 = 10$ . Note that when  $m_1 = 1$ , MT-EM and MT-EM-Bag are respectively equivalent to the Chow-Liu algorithm and MT-Bag (with  $m = m_2 = 10$ ). We outline the results of those methods by two constant lines on the figures. Those results were obtained on 5 networks times 6 data sets for each  $N$ .

At 120 samples the Chow-Liu algorithm is better than MT-EM: because of the low number of samples, the negative impact of an increased variance is not compensated by the gain in bias when increasing  $m_1$ . Using MT-EM-Bag rather than MT-EM (with  $m_1 = 2$ ) leads to a better accuracy than Chow-Liu, but is worse than MT-Bag alone. The two-level mixtures can not completely compensate for the lack of samples.

As the number of samples increases, the variance of all models decreases, and the smaller bias of larger mixtures progressively gives them the advantage over Chow-Liu. Constructing an optimal mixture of size 2 is more interesting than a Chow-Liu tree for  $N = 2000$  samples. At this point however, reducing the variance of Chow-Liu is still more interesting since MT-Bag is better than MT-EM for all  $m_1$ .

As more and more samples are available, the variance becomes even lower (MT-Bag is closer to Chow-Liu), and reducing the bias becomes

preferable to lowering the variance. At 8000 samples, MT-EM is definitely better than MT-Bag and the KL divergence decreases when  $m_1$  increases. Nevertheless, MT-EM-Bag is still better than MT-EM, showing that reducing the variance of each term is still interesting.

Moreover, the gap between MT-EM and MT-EM-Bag is widening as  $m_1$  increases. A larger  $m_1$  means building each second-level mixture on fewer samples, increasing the variance, a situation favoring it over a single Chow-Liu tree.

## 6 Conclusion

In this work we propose to incorporate the perturb and combine variance reduction approach within the maximum likelihood approach to learn mixtures of Markov trees. The resulting algorithms build a two-level mixture of Markov trees, by replacing each term (a Chow-Liu tree) of a mixture produced by the EM algorithm by a mixture of bagged Markov trees.

The resulting models are shown to improve those constructed by the base EM algorithm. The performance advantage of the two-level mixture of Markov trees over the single level EM mixture increases with the number  $m_1$  of terms used in the upper level of the mixture, even if at very low sample size increasing  $m_1$  may be detrimental in general (and hence a simple bagged ensemble of Chow-Liu trees is then better). Although we do not study in detail

the effect of the parameter  $m_2$ , the lower level mixture size, increasing this number can only further improve the advantage of the two-level methods with respect to the single level one.

Several questions remain to be studied. We replace each term by a mixture only after the convergence of the EM algorithm by fear of disrupting its convergence properties and for complexity reasons. Could doing so earlier further limit overfitting?

The number  $m_2$  of terms in the randomized sub-mixture is always fixed to 10. Its effect could be further studied with respect to a possible accuracy/inference time trade-off. In particular, since the number of samples influences the gap between the Chow-Liu tree and the mixture of bagged Markov trees, should this parameter differ for each term of the EM mixture, based on the number of samples associated to that term?

## Acknowledgments

François Schnitzler is supported by a F.R.I.A. scholarship. This work was also funded by the Biomagnet IUAP network of the BELSPO and the Pascal2 network of excellence of the EC. The scientific responsibility is the authors'.

## References

- S. Ammar, P. Leray, F. Schnitzler, and L. Wehenkel. 2010. Sub-quadratic Markov tree mixture learning based on randomizations of the Chow-Liu algorithm. In *5th European Workshop on Probabilistic Graphical Models*, pages 17–24.
- V. Auvray and L. Wehenkel. 2008. Learning inclusion-optimal chordal graphs. In *24th UAI*, pages 18–25.
- F. R. Bach and M. I. Jordan. 2001. Thin junction trees. In *Advances in Neural Information Processing Systems 14*, pages 569–576. MIT Press.
- L. Breiman. 1996. Arcing classifiers. Technical report, Dept. of Statistics, University of California.
- L. Breiman. 1999. Using adaptive bagging to debias regressions. Technical report, Dept. of Statistics, University of California.
- D.M. Chickering, D. Geiger, and D. Heckerman. 1994. Learning Bayesian networks is NP-hard. Technical report, Microsoft Research.
- C.I. Chow and C.N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory*, 14:462–467.
- G.F. Cooper. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38.
- G. Elidan and S. Gould. 2008. Learning bounded treewidth Bayesian networks. *JMLR*, 9:2699–2731.
- S. Kirshner and P. Smyth. 2007. Infinite mixtures of trees. In *24th International Conference on Machine Learning*, pages 417–423.
- D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models*. MIT Press.
- J. Kollin and M. Koivisto. 2006. Bayesian learning with mixtures of trees. *ECML*, pages 294–305.
- J. Kwisthout, H. Bodlaender, and L. van der Gaag. 2010. The necessity of bounded treewidth for efficient inference in Bayesian networks. In *19th European Conference on Artificial Intelligence*, pages 623–626.
- H. Liu, M. Xu, A. Gupta H. Gu, J. Lafferty, and L. Wasserman. 2011. Forest density estimation. *JMLR*, 12:907–951.
- G.J. McLachlan and T. Krishnan. 2008. *The EM Algorithm and Extensions*, volume 382. John Wiley and Sons.
- M. Meila and T. Jaakkola. 2006. Tractable Bayesian learning of tree belief networks. *Statistics and Computing*, 16:77–92.
- M. Meila and M.I. Jordan. 2001. Learning with mixtures of trees. *JMLR*, 1:1–48.
- J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- 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*, pages 219–224.
- V. Y.F. Tan, A. Anandkumar, and A. S. Willsky. 2011. Learning high-dimensional Markov forest distributions: Analysis of error rates. *JMLR*, 12:1617–1653.

# Graphical inequality diagnostics for phylogenetic trees

Nathaniel Shiers

University of Warwick, UK

n.l.shiers@warwick.ac.uk

Jim Q. Smith

University of Warwick, UK

j.q.smith@warwick.ac.uk

## Abstract

Phylogenetic trees can be studied as discrete graphical models with interior vertices hidden and leaves observed. In the case of binary random variables it has long been known that this class of models constrains the parameter space of the leaf margin. However, it was only last year that the inequality constraints were fully characterised for binary phylogenetic trees. The inferential consequences of these implicit inequality constraints have so far attracted little attention. However, this paper uses a subset of these constraints to demonstrate their usefulness as simple and transparent diagnostics that can be used at the preliminary stages of a phylogenetic analysis, not only to check whether any phylogenetic model is consistent with the data but also to guide the ensuing inference.

## 1 Review of implicit constraints

The estimation of Bayesian networks when interior vertices are hidden is challenging since the implicit geometry of the associated probability mass function of the observed variables, and hence also the likelihood, is complicated.

The simplest such model is the phylogenetic tree where all variables are binary. A phylogenetic tree is a graphical method for describing the evolutionary relationships between species both extant and extinct. The geometry of this space of models was expanded in Settimi and Smith (2000) and has recently attracted considerable interest (for example Allman et al. (2009) and Zwiernik and Smith (2012)). These advances have encouraged some authors to proceed to use the known geometry of these spaces to support inference and learning over the space of tree models (see Drton and Sullivant (2007) for example). These focus on the polynomial constraints that are implicit in these models. However, it is well known that not only these functional relationships but also additional inequality constraints

are active. Last year Zwiernik and Smith (2011) fully characterised these inequality constraints for binary phylogenetic trees. So we are at last able, at least for this important class of graphical models, to explore the inferential use for learning of these derived inequality constraints.

In this paper we demonstrate how some of these constraints can be used for inference by providing the basis for various diagnostics. These are primarily designed for the early stages of a phylogenetic analysis. The first point we note is that all trees must satisfy certain cubic inequalities associated with all the triples of its observed variables. A simple diagnostic is produced which therefore simply checks whether data appears to fit with any tree structure or whether some of these inequalities appear to be violated. For if these constraints are significantly broken then we know that no tree will fit the data well and a search for a more elaborate graphical model of the data is needed. We also indicate how functions of the associated statistics can be used as a guide to which candidate tree models are likely to score highly in model selection. In this way, these preprocessing tools can be used to

help speed up and stabilise numerical scoring methods over the class.

A phylogenetic tree is often studied as the pattern (see Verma and Pearl (1990)) of a particular class of Bayesian networks, specifically trees (e.g. Dutkowska and Tiuryn (2007)). This paper will consider trees labelled by binary random variables each taking a value of 0 or 1 (for the results we use, we only require that the observed variables are binary). The variables on the trees considered will be classified into two sets: the  $m - 2$  hidden nodes  $H \in \mathcal{H}$  and  $m$  manifest nodes  $X \in \mathcal{X}$  (denoted by white and black circles respectively). We will consider trees with manifest variables as the leaves and all remaining variables hidden (i.e. never directly observed). It was shown in Settimi and Smith (2000) that all phylogenetic tree models are contained in the class of strictly trivalent trees, that is trees where all hidden vertices have degree 3. The results of this paper therefore focus on this class. To relate a trivalent tree to a phylogenetic tree rooted on a selected hidden node, a further hidden node can be inserted to produce a bifurcating tree from the root whilst remaining Markov equivalent. In Settimi and Smith (2000) it is shown that all the conditional independence statements implicit in a tree in terms of algebraic functions in their moments can be equivalently re-expressed in terms of conditions on moments. However, it is the novel parametrisation to tree cumulants by Zwiernik and Smith (2012, Section 3)) that allowed the authors to specify explicitly and elegantly all the constraints and conditions imposed by a strictly trivalent tree structure (Zwiernik and Smith (2011, Theorem 4.7)).

### 1.1 Constraints on a tripod tree

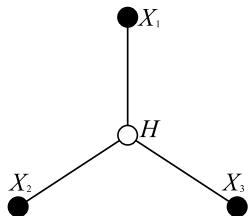


Figure 1: Tripod tree.

The inequality constraints imposed by the

implicit conditional independence on the tripod tree with binary random variables  $X_1, X_2, X_3$  (Figure 1), were originally given in Settimi and Smith (1998). Here we adopt the equivalent formulation described in Zwiernik and Smith (2011, Proposition 2.5). Using the definition of  $k^{\text{th}}$  central moment as  $\mathbb{E}[(X - \mathbb{E}[X])^k]$  we denote the applicable central moments as  $\mu_{12}, \mu_{13}, \mu_{23}, \mu_{123}$  (equivalent to tree moments for order 2 and 3), and the means as  $\lambda_1, \lambda_2, \lambda_3$ . Then given an observed joint probability table  $P$  ( $2 \times 2 \times 2$ ), the data is consistent with the tripod tree structure if and only if:

- (i)  $\mu_{123} = 0$  and at least two of the three covariances  $\mu_{12}, \mu_{13}, \mu_{23}$  vanish or
- (ii)

$$\mu_{12}\mu_{13}\mu_{23} > 0 \quad (1.1)$$

$$|\mu_{jk}| \sqrt{\det(P)} + \mu_{123}\mu_{jk} \leq (1 + \bar{\mu}_i)\mu_{jk}^2 \quad (1.2)$$

$$|\mu_{jk}| \sqrt{\det(P)} - \mu_{123}\mu_{jk} \leq (1 - \bar{\mu}_i)\mu_{jk}^2 \quad (1.3)$$

where  $\det(P) = \mu_{123}^2 + 4\mu_{12}\mu_{13}\mu_{23}$  and  $\bar{\mu}_i = 1 - 2\lambda_i$  for all  $i \in \{1, 2, 3\}$ . We can view these admissible regions by fixing  $\mu_{123}$  and  $\bar{\mu}_i$ , and then setting the covariances as the axes in three-dimensional space. Fixing  $\bar{\mu}_i = 0.5$  for all  $i$  produces the largest regions of tree admissible values of moments (all else equal). We can then view separate plots for different values of  $\mu_{123}$ , namely 0, 0.01, 0.02, 0.03 (see Figure 2). It is clear that as  $\mu_{123}$  increases the admissible regions implied by the tree structure decrease. In fact, in the example above, as  $|\mu_{123}|$  increases towards 0.03125 the region reduces to 4 singular points, and beyond this disappears.

The boundary points of these graphical regions typically represent zeros in the marginal tables of each pair comprising a manifest node and  $H$ . This is equivalent to having a manifest node equal to  $H$ . In the space of the three second-order central moments, the edges represent having zero entries in two tables, and each vertex corresponds to zeros in all three tables. Thus, the boundary points of the geometry of the parameter spaces all relate to varying degrees of degenerate distributions. These degeneracies in hidden variable models have been studied for some time and it is understood that when data does not obey the conditional independence

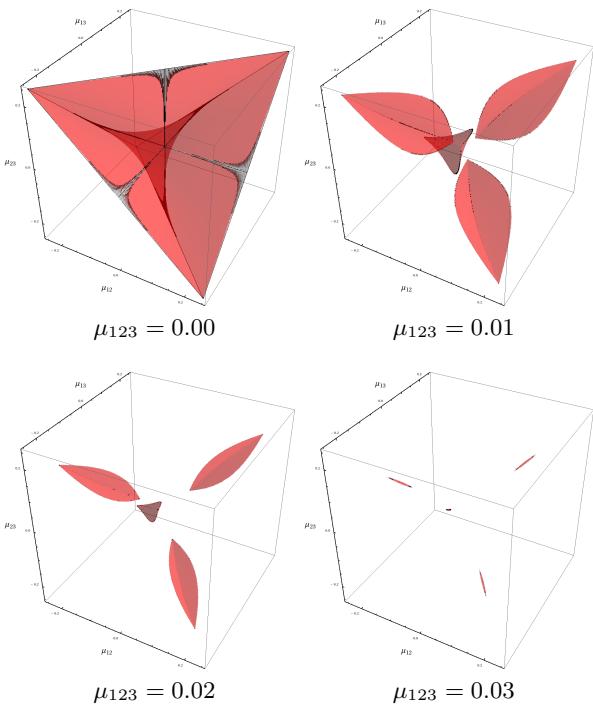


Figure 2: Admissible regions of a tripod tree.

assumptions of a tree structure, the model with highest likelihood will contain some zeroes. This in turn implies particular functional relationships between the variables must hold (see, for example, de Leeuw et al. (1990)).

It is important to state that if the marginal data on the manifest nodes is inconsistent with a tree model, then in the limit (consistent) sample covariance estimates of the the manifest variables will correspond to points outside of these feasible regions. The geometries of such admissible regions are later illustrated in Section 3.

## 2 Two useful graphical results

The following theorem which has appeared previously, albeit as a corollary of more general theory in different forms and different contexts, has yet to be exploited but is particularly useful:

**Theorem 1.** *The constraints we can use for a tree diagnostic can be derived simply in the following graphical way:*

*Given any strictly trivalent phylogenetic tree  $T$ , for all triples  $X_i, X_j, X_k$  there exists a unique hidden variable  $H_{ijk}$  such that  $H_{ijk}$  separates  $(X_i, X_j, X_k)$  in  $T$ . i.e.  $\perp\!\!\!\perp (X_i, X_j, X_k)|H_{ijk}$*

See Appendix A for proof.

This result allows us to construct a diagnostic test to check whether any tree could be consistent with a sample data set (see Section 4). This method is not the only means of assessing tree structures (e.g. the retention index Farris (1989)) but has the advantage of being very simple to implement and being grounded in theory. When the diagnostic does not reject the tree class, there is a second way in which the distributions of triples can be used to guide the search for promising candidate models.

First note that, by its definition, associated with every hidden variable  $H \in \mathcal{H}$  of a strictly trivalent tree  $T$  is a partition  $\Lambda(H, T)$  of the manifest variables into 3 subsets, each subset being the leaves of a subtree rooted at  $H$ . Interestingly, these partitions uniquely define a tree  $T$ . Thus we have:

**Theorem 2.** *Each strictly trivalent tree  $T$  is uniquely identified by its set of partitions and  $\mathcal{X}(T) \triangleq \{\Lambda(H, T) : H \in \mathcal{H}\}$  acts as an identifier, under the assumption of faithfulness (see Spirtes et al. (2001)).*

See Appendix B for proof.

Thus if there exists a unique phylogenetic tree, then the estimated moments of the triple will identify it. We will show that this simple result allows us to preselect good candidate trees for model selection. See Section 5 for more details.

## 3 An illustration of the constraints

In this section we simulated binary data from the two non-isomorphic strictly trivalent trees with 6 leaves. Then from two other graphical models chosen to mirror typical variations which, for scientific reasons, we might expect of the tree. Unsurprisingly the empirical moments derived from the tree-generated data satisfy all the constraints whilst the empirical moments calculated from the non-tree data violates some of the constraints. The four graphs are shown in Figure 3.

The variations of the two trees were chosen to be similar in order to highlight any other differences. The probability distributions of the graphs were simulated so expected means for each  $X_i$  were the same across graphs, thus the

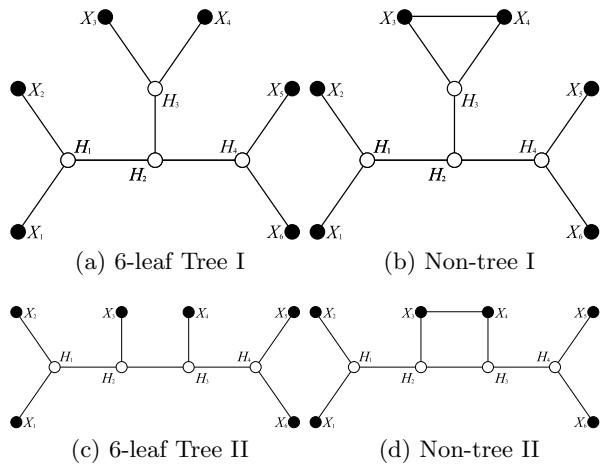


Figure 3: 6-leaved trees and non-trees.

differences in the graphs may be expected to be highlighted in the higher order moments.

For large enough samples, the sample moments of the trees always satisfied the constraints demanded of their population analogues whilst the non-tree modifications did not. Both non-trees experienced the same violations:  $(X_1, X_3, X_4)$  and  $(X_2, X_3, X_4)$  for Inequality (1.2),  $(X_3, X_4, X_5)$  and  $(X_3, X_4, X_6)$  for Inequality (1.3), and all versions of Inequality (1.1) which involve  $X_{34}$  (e.g.  $X_{13}X_{14}X_{34} < 0$ ). Note that all violated constraints include both indices 3 and 4 (where the extra edge was added). Moreover, every inequality involving indices 3 and 4 is broken. This suggests that these diagnostics could provide more insight than a binary acceptance or rejection of the tree structure; the violated constraints in these cases hint that  $X_3$  and  $X_4$  may be responsible for the exceptions.

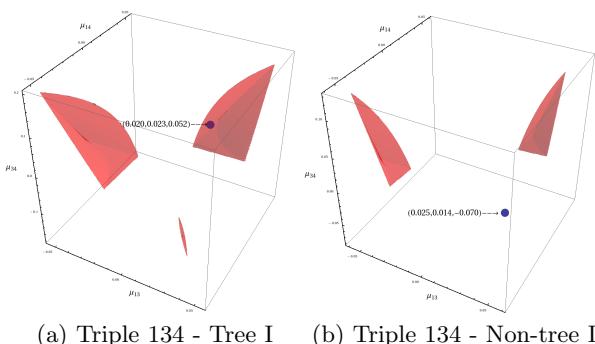


Figure 4: Plots of covariance point estimates.

The geometries of the admissible regions of the

covariances  $\mu_{13}, \mu_{14}, \mu_{34}$  are similar between the trees, and similar between the non-trees. The graphs of the admissible covariance regions are shown for Figure 3a and Figure 3b in Figure 4a and Figure 4b respectively, along with the estimates of the observed covariances plotted. The sample covariances are revisited in Section 6.

#### 4 Application of diagnostics

Phylogenetic trees can be constructed using gene sequences where the vectors of data are obtained from DNA sequences coded into binary. Each base in a sequence has one of four chemicals either T or C (pyrimidines) or A or G (purines) (Yang (2007)). We thus encode T and C as 1, and A and G as 0. To illustrate the use of diagnostics we use mitochondrial genetic data each of length 883 base pairs from BOLD Systems 3 (<http://boldsystems.org>) for six species from the class Mammalia:

$X_1$ —*Ailurus fulgens* (red panda)

$X_2$ —*Procyon lotor* (raccoon)

$X_3$ —*Ailuropoda melanoleuca* (giant panda)

$X_4$ —*Ursus maritimus* (polar bear)

$X_5$ —*Tremarctos ornatus* (spectacled bear)

$X_6$ —*Ursus malayanus* (sun bear)

We here assume that the selected region of DNA is appropriate and use the sampled moments to check whether any tree structure may be appropriate given this data.

The values of the constraints were calculated using the data to estimate the parameters, where crude sample estimates were used. Bahadur expansions can be used to produce estimators of the higher order terms (see Streitberg (1990)).

The genetic data did violate some constraints for five of the triples:  $(X_4, X_6, X_k)$  for  $k = \{1, 2, 3, 5\}$ , and  $(X_2, X_4, X_5)$ . Figure 5a shows a covariance triple point within the admissible region, and contrastingly Figure 5b shows a different sample covariance triple outside the region associated with a tree. Superficially, this suggests that a tree is not a suitable model for the data. However, because the sample size is not large these violations could be attributed to random error, particularly given the closeness of the point in Figure 5b to the boundary of the admissible region. This is explored in Section 5.

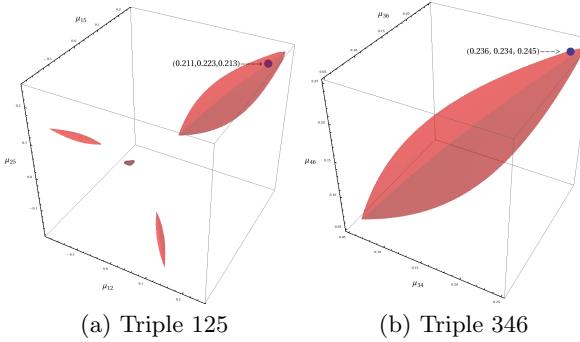


Figure 5: Point estimates of covariances.

## 5 The effect of sample size

We now examine the likely power of these diagnostics for studies like those given below using data generated from the graphs in Figure 3 for smaller samples. Since the structure of the graphs is known, for the trees any violations of the constraints are due to estimation error of the moments. The inequalities can be tested for each of the four graphs using different sample sizes ( $n = 500, 883, 1500, 5000$ ), and repeated  $10^4$  times. For the 20 combinations of triples on each graph the number of triples which violate at least one constraint is recorded. This is relevant as even for a tree, some violations (depending on the sample size) will be expected due to noise. A comparison of these results is shown in Figure 6 for Tree I and Non-tree I from Figure 3.

It is noticeable at a sample size of 883 (as in Section 4) there is a clear shift between the modal violations in the non-tree model relative to the tree model. If 5 or fewer violations are observed Tree I seems the more likely model, and if 11 or more then Non-tree I appears the favourable hypothesis. Note that the frequencies for the non-tree are likely to be conservative (shifted to the left) compared to more complex non-trees, and so the comparison is also prudent.

We compared the two six-leaved tree models and noted that they performed similarly in terms of number of violations. The sample size required for these diagnostics to be useful is likely to vary depending on the topology of the models, as well as the underlying true joint probability table. In the examples we have looked at, a large number of violations would suggest a non-tree

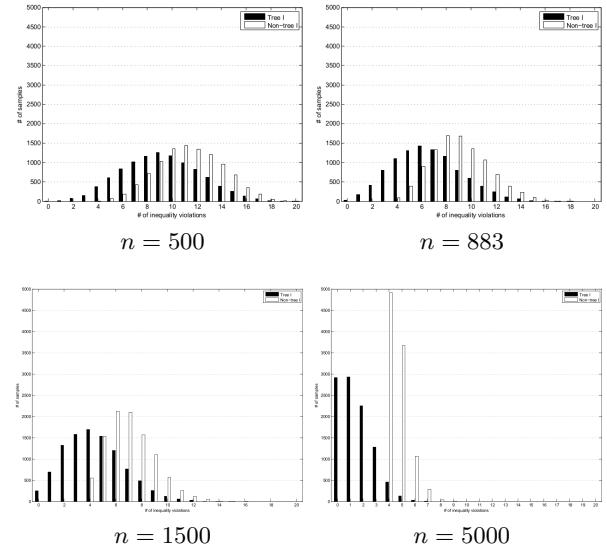
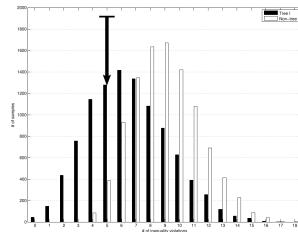


Figure 6: Frequency of violations on a tree (black) and non-tree (white) for 4 sample sizes.

model even for a moderate sample size.

Returning to the application in Section 4 where 5 violations were observed, the frequency of violation plots can be studied. Figure 7 shows the plot for Tree I and Non-tree I, where the arrow indicates the frequencies for 5 violations. If the plot is thought to be typical of other distributions on trees of this size, the level of violations would suggest that the gene sequence data is more likely to conform to a tree than a non-tree. The compared non-tree is from the simplest subset of non-trees (only one additional edge). A more complex tree would be expected to produce more violations and so additional weight may be given to the hypothesised tree.

Figure 7: Frequency of violations ( $n = 883$ ).

## 6 Inferring trees from moments

It can be shown that the first 3 moments provide us with consistent but inefficient estimates of a tree. In Settimi and Smith (1998) it was shown

that (provided none of the terms is zero) for any triple  $X_i, X_j, X_k$  with  $(X_i \perp\!\!\!\perp X_j \perp\!\!\!\perp X_k) | H_{ijk}$   $S_{ijk} = \ln(|\mu_{ij}|) + \ln(|\mu_{ik}|) + \ln(|\mu_{jk}|) - 2 \ln(|\mu_{ijk}|)$  where  $S_{ijk}$  (the signature) depends only on the margin distributions of the triple. From Theorem 2, for large enough datasets the signatures of the corresponding sample quantities will indicate candidate tree partitions  $\mathcal{X}(\mathcal{T})$  and hence, from the theorem, candidates  $\mathcal{T}$ . Note that triples  $(i, j, k)$  and  $(i', j', k')$  share the same separator  $H$  in  $\mathcal{T}$  if the pairs  $(i, i')$ ,  $(j, j')$  and  $(k, k')$  all lie in different subsets in  $\Lambda(H, \mathcal{T})$ . So these  $(n-2)$  partitions can be calculated by first clustering the signatures by magnitude into up to  $(n-2)$  clusters and from this deducing  $\Lambda(H, \mathcal{T})$ . For small trees this method, simply using the statistics already calculated for our first diagnostics, allows us to identify some promising trees.

Figure 8 shows the standardised signatures  $S_{ijk}^*$  for both trees in Figure 3 with the signatures of interest labelled. The clustering of the signatures demonstrates the power of the diagnostic - there is remarkably clear separation of all  $S_{12k}^*$  and  $S_{i56}^*$  which supports the topologies of the trees. The signatures involving 3 and 4 are less distinct, but this may be in part unavoidable overlapping of true clusters. However, trees like Tree I have an interior node and it can be shown that this leads to signatures with a higher variance. The wide spread in Tree I of the middle cluster (overlapping) with the  $X_3 X_4$  cluster is indicative of this.

When applied to the genetic data we get strong clustering of signatures involving the raccoon and giant panda, and some clustering of polar bear and sun bear - the signatures for the remaining species (red panda and spectacled bear) are dispersed. This diagnostic would thus suggest Tree II as a starting point for an analysis fitting the data, with the latter species being the singletons.

Finally, the sample covariances  $\hat{\mu}_{ij}$  themselves can provide some indication of the topology of the tree if correlations between each manifest variable and its hidden parent are of about the same magnitude. For then  $\hat{\mu}_{ij}$  tends to be higher when the number of edges between these vertices is fewer (as utilised in

Harmeling and Williams (2011)). For example, considering Section 3, for Tree I  $\hat{\mu}_{56}$ ,  $\hat{\mu}_{12}$  and  $\hat{\mu}_{34}$  have the greatest absolute values and for Tree II  $\hat{\mu}_{56}$  and  $\hat{\mu}_{12}$  have greatest magnitude, but now  $\hat{\mu}_{23}$  and  $\hat{\mu}_{24}$  rank above  $\hat{\mu}_{34}$  in magnitude. This appears to reflect the structure of Tree II where  $X_3$  and  $X_4$  are not directly joined to the same vertex.

It follows that non-metric multidimensional scaling (MDS) can be used on a function of sample covariances, allowing the relationships to be displayed graphically. Here we use the function  $\delta_{ij} = -2 \ln(|\hat{\mu}_{ij}|)$

The resulting plots (Figure 9) for a 2D scaling relate to the trees in Figure 3. They were generated using a sample sizes of 883, which gives a relevant comparison with the gene data (see Figure 10). Inequality violation is expressed through the size of the plotted points. The relative sizes are determined via  $(1 + V_i)^{-1/2}$  where  $V_i$  is the number of violations for variable  $X_i$ .

For Tree I, the MDS indicates clear pairings  $(X_1, X_2)$ , and  $(X_5, X_6)$ , plus  $X_3$  and  $X_4$  are reasonably close which suggests these three pairs are a distance of 2 edges apart giving us the topology here. However in Tree II  $X_3$  and  $X_4$  are further apart so might be conjectured as singletons - (giving us Tree II). Note that this ambiguity may be caused in part by some  $|\hat{\mu}_{ij}|$  being close to zero.

Figure 10 is the result of performing the same MDS for the genetic data with the hope that the plot indicates one of the two six-leaved trees. Unlike the previous plots, the points are separated into two groups of three. This matches the form of Tree II, and although some of the distances are similar it could be hypothesised that  $(X_2, X_4)$ , and  $(X_5, X_6)$  are pairs, with  $X_1$

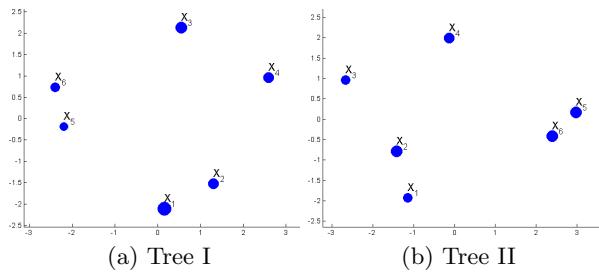


Figure 9: MDS for trees in Figure 3.

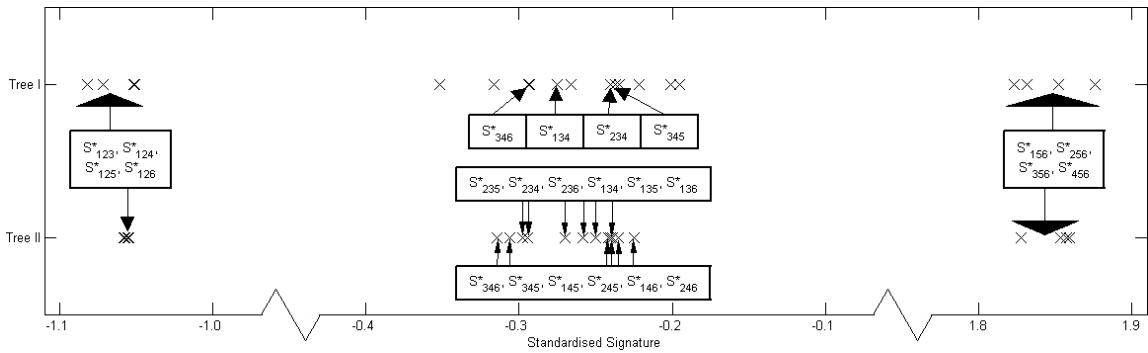


Figure 8: Plot of standardised signatures for Tree I and Tree II.

and  $X_3$  as singletons.

It is interesting to note that the ambiguity is reflected in a more detailed analysis of this data set. We are currently using these simple methods described to preselect good trees with the hope of achieving a large time saving at little cost to accuracy. We then use the subset of trees as a starting point for MCMC likelihood-based model selection algorithms, which otherwise often get stuck within local maxima (e.g. see Chor et al. (2000)).

## 7 Discussion

In this paper we have illustrated how some simple graphical properties of trees allow us to construct useful diagnostics based on inequality violations and certain functions of sample moments up to degree 3. The diagnostics are trivial to calculate and for the sizes of data sets used in phylogenetic trees (now typically 1500 base pairs if using BOLD) provide a relatively powerful method. In particular the inequality diagnostics are complementary to the algebraic methods developed by Drton and Sullivant (2007) which are based on different functional constraints. We are currently developing analogous inequality diagnostics using the same graphical properties

but for differently distributed variables.

One appealing and unusual feature of our methods based on low order moments is that as we add more species to the putative tree we simply need to check the new triples introduced by the additional manifest variables. So in this sense it scales up. Furthermore if violations of the tree structure are discovered our methods also sometimes allow us to identify a subset of manifest variables on which a tree is valid. So for example in the two non-trees of Figure 3 we can still deduce that a tree might be valid on  $X_1, X_2, X_5, X_6$  since no violations of the inequality constraints are apparent.

Of course, rather than simple singularities it is important to develop more general theory for tree diagnostics so that they can be exploited routinely. But even in this naive form, our methods appear surprisingly effective. This work is continuing and we will report our findings in a future paper.

We gratefully acknowledge the referees for their thorough and valuable feedback.

## References

- E S Allman, C Matias, and J A Rhodes. 2009. Identifiability of parameters in latent structure models with many observed variables. *Ann. Stat.*, 37(6A):3099–3132.
- B Chor, M D Hendy, B R Holland, and D Penny. 2000. Multiple maxima of likelihood in phylogenetic trees: an analytic approach. *Mol. Biol. Evol.*, 17(10):1529–1541.
- J de Leeuw, P G M van der Heijden, and P Ver-

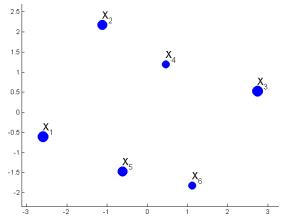


Figure 10: MDS for gene data.

- boon. 1990. A latent time-budget model. *Stat. Neerlandica*, 44(1):1–22.
- M Drton and S Sullivant. 2007. Algebraic statistical models. *Stat. Sinica*, 17(4):1273–1297.
- J Dutkowski and J Tiuryn. 2007. Identification of functional modules from conserved ancestral protein-protein interactions. *Bioinforma.*, 23(13):149–158.
- J S Farris. 1989. The retention index and the rescaled consistency index. *Cladistics*, 5(4):417–419.
- S Harmeling and C K I Williams. 2011. Greedy learning of binary latent trees. *Pattern Anal. Mach. Intell.*, 33(6):1087–1097.
- R Settimi and J Q Smith. 1998. On the geometry of bayesian graphical models with hidden variables. In *Proc. 14th Conf. Uncertain. Artif. Intell.*, pages 472–479. Morgan Kaufmann.
- R Settimi and J Q Smith. 2000. Geometry, moments and conditional independence trees with hidden variables. *Ann. Stat.*, 28(4):1179–1205.
- P Spirtes, C Glymour, and R Scheines. 2001. *Causation, Prediction, and Search*. MIT Press, 2nd edition.
- B Streitberg. 1990. Lancaster interactions revisited. *Ann. Stat.*, 18(4):1878–1885.
- T Verma and J Pearl. 1990. Equivalence and synthesis of causal models. In *Proc. 6th Conf. Uncertain. Artif. Intell.*, pages 255–270. Elsevier Science.
- Z Yang. 2007. *Computational Molecular Evolution*. Oxford University Press.
- P Zwiernik and J Q Smith. 2011. Implicit inequality constraints in a binary tree model. *Electron. J. Stat.*, 5:1276–1312.
- P Zwiernik and J Q Smith. 2012. Tree cumulants and the geometry of binary tree models. *Bernoulli*, 18(1):290–321.

## Appendix A Proof of Theorem 1

*Proof.* Let  $X_i, X_j, X_k$  be any three manifest variables (leaves) on a phylogenetic tree  $T$ . Let  $X_a-X_b$  denote a path between  $X_a$  and  $X_b$ . Similarly,  $X_a-X_b-X_c$  denotes a path between  $X_a$  and  $X_c$ , and the path contains at least  $X_b$ .

By properties of a tree, there is exactly one path with no repeated edges  $X_i-X_j$  and similarly one such path  $X_i-X_k$  and  $X_j-X_k$ . Note that the

intersection of the non-repeating paths  $X_i-X_j$  and  $X_i-X_k$  has at least one hidden vertex as it contains the hidden node adjacent to  $X_i$ . Denote the vertex in this intersection furthest from  $X_i$  as  $H^*$ , thus the intersection of  $X_j-H^*$  and  $X_k-H^*$  is the vertex  $H^*$ . Now consider the repeating path  $X_j-H^*-X_i-H^*-X_k$ . By removing the repeating nodes and their edges, a non-repeating subpath  $X_j-H^*-X_k$  is formed, with  $H^*$  being the only remaining node from the intersection of  $X_i-X_j$  and  $X_i-X_k$ . Thus  $H^* = H_{ijk}$  and furthermore  $H_{ijk}$  is unique as no other vertex appears on all three non-repeating paths  $X_i-X_j$ ,  $X_i-X_k$  and  $X_j-X_k$ .  $\square$

## Appendix B Proof of Theorem 2

*Proof.* Let  $H'$  be a vertex in  $\mathcal{T}$  which is a leaf of the subtree  $\mathcal{T}^H$  consisting of all hidden vertices and their connecting edges in  $\mathcal{T}$ . Then since  $\mathcal{T}$  is strictly trivalent and  $H'$  is an interior vertex in  $\mathcal{T}$ ,  $H'$  must be connected to two manifest vertices of  $\mathcal{T}$  which we label  $X_{m-1}$  and  $X_m$ . Thus  $\{X_{m-1}\}$  and  $\{X_m\}$  are singletons in  $\Lambda(H', \mathcal{T})$ .

Suppose there exists  $m$ , the lowest number of leaves a strictly trivalent tree can have such that there exists nonisomorphic  $\mathcal{T}_1$  and  $\mathcal{T}_2$  with  $\mathcal{X}(\mathcal{T}_1) = \mathcal{X}(\mathcal{T}_2)$ . For  $m = 3$  there is only one strictly trivalent tree, so  $m \geq 4$ . Now select  $H' \in \mathcal{H}$  such that  $\Lambda(H', \mathcal{T}_1)$  ( $= \Lambda(H', \mathcal{T}_2)$ ) contains observed vertices  $X_{m-1}, X_m$  as singletons.

Note then that for all other  $H \in \mathcal{H}$  by the definition of  $\mathcal{X}(\mathcal{T}_1)$  (and  $\mathcal{X}(\mathcal{T}_2)$ ) the pair  $\{X_{m-1}, X_m\}$  are contained in the same subset of both partitions  $\Lambda(H, \mathcal{T}_1)$  and  $\Lambda(H, \mathcal{T}_2)$ . So

$\mathcal{X}(\mathcal{T}') = \{\Lambda(H, \mathcal{T}_i) : H \in \mathcal{H} \setminus \{H'\}, i = 1, 2\}$  is isomorphic to a set of partitions of  $\{X_1, \dots, X'_{m-1}\}$  where  $X'_{m-1}$  is identified with  $\{X_{m-1}, X_m\}$ .

Now define trees  $\mathcal{T}'_i$  from  $\mathcal{T}_i$ ,  $i = 1, 2$  each having  $m - 1$  observed vertices: In  $\mathcal{T}'_i$  replace  $H' \in \mathcal{H}$  by a manifest variable  $X'_{m-1}$  then delete vertices  $X_{m-1}, X_m$  and their connecting edges.

By construction  $\mathcal{X}(\mathcal{T}'_1) = \mathcal{X}(\mathcal{T}'_2)$ , yet  $\mathcal{T}'_1$  and  $\mathcal{T}'_2$  have  $m - 1$  manifest variables so by the definition of  $m$  they must be isomorphic. But then by construction  $\mathcal{T}_1$  and  $\mathcal{T}_2$  are also isomorphic. Thus no such  $m$  exists. Contradiction.  $\square$

# Learning Multivariate Regression Chain Graphs under Faithfulness

Dag Sonntag

ADIT, IDA, Linköping University, Sweden

dag.sonntag@liu.se

Jose M. Peña

ADIT, IDA, Linköping University, Sweden

jose.m.pena@liu.se

## Abstract

This paper deals with multivariate regression chain graphs, which were introduced by Cox and Wermuth (1993, 1996) to represent linear causal models with correlated errors. Specifically, we present a constraint based algorithm for learning a chain graph a given probability distribution is faithful to. We also show that for each Markov equivalence class of multivariate regression chain graphs there exists a set of chain graphs with a unique minimal set of lines. Finally, we show that this set of lines can be identified from any member of the class by repeatedly splitting its connectivity components according to certain conditions.

Keywords: Chain Graph, Multivariate Regression Chain Graph, Learning, Bidirected Graph

## 1 Introduction

In this paper we deal with multivariate regression chain graphs (CGs) which were introduced by Cox and Wermuth (1993, 1996). Graphically Cox and Wermuth represent these CGs with dashed edges to distinguish them from other interpretations, e.g. LWF (Lauritzen, 1996) or AMP (Andersson et al., 2001). Multivariate regression CGs also coincide with the acyclic directed mixed graphs without semi-directed cycles presented by Richardson (2003). A fourth interpretation of CGs can also be found in Drton (2009). The different interpretations of CGs have different merits, but none of the interpretations subsumes another interpretation (Drton, 2009).

The multivariate regression CG interpretation still misses some fundamental elements found and proven in the LWF and AMP interpretations. In this article we study and describe two of these elements. The first is a learning algorithm for learning a CG from a proba-

bility distribution faithful to a multivariate regression CG. Similar algorithms have been presented for the LWF (Studený, 1997; Ma et al., 2008) as well as the AMP (Peña, 2012) interpretations. The algorithm presented is constraint based and resembles both Studený's and Peña's algorithms. The second element we present is a feasible split operation and a feasible merge operation similar to the ones presented for the LWF and AMP interpretations (Studený et al., 2009). These splits and mergings can be used to alter the structure of a multivariate regression CG in such a way that it does not change the CG's Markov equivalence class. Finally we show that for each Markov equivalence class of multivariate regression CGs there exists a set of CGs with a unique minimal set of lines. We also show that this set of CGs can be reached by applying feasible splits to any CG in the same Markov equivalence class.

The rest of the paper is organised as follows. Section 2 reviews the concepts used in the rest of the article. Section 3 presents the feasible split and merge operations. Section 4 presents the learning algorithm and proves its correctness.

Section 5 closes the article with some discussion.

## 2 Preliminaries

In this section, we review some concepts from probabilistic graphical models that are used later in this paper. All the graphs and probability distributions in this paper are defined over a finite set of variables  $V$ . With  $|V|$  we mean the number of variables in the  $V$  and with  $|V_G|$  the number of variables in the graph  $G$ . Throughout the paper the intended meaning of CGs is multivariate regression CGs if no other interpretation is mentioned. To allow more readable figures bidirected edges are used instead of dashed edges or lines. To not confuse the reader these edges will also be denoted bidirected edges throughout the article.

If a graph  $G$  contains an edge between two nodes  $V_1$  and  $V_2$ , we write that  $V_1 \rightarrow V_2$  is in  $G$  for a directed edge,  $V_1 \leftrightarrow V_2$  is in  $G$  for a bidirected edge, and  $V_1 - V_2$  is in  $G$  for an undirected edge. With  $V_1 \rightsquigarrow V_2$  we mean that either  $V_1 \rightarrow V_2$  or  $V_1 \leftrightarrow V_2$  is in  $G$ . With  $V_1 \multimap V_2$  we mean that either  $V_1 \rightarrow V_2$  or  $V_1 - V_2$  is in  $G$ . With  $V_1 \rightsquigarrow V_2$  we mean that there exists an edge between  $V_1$  and  $V_2$  in  $G$ . A set of nodes is said to be complete if there exists edges between all pairs of nodes in the set. A complete set of nodes is said to be a clique if there exists no superset of it that is complete.

The parents of a set of nodes  $X$  of  $G$  is the set  $pa_G(X) = \{V_1 | V_1 \rightarrow V_2 \text{ is in } G, V_1 \notin X \text{ and } V_2 \in X\}$ . The children of  $X$  is the set  $ch_G(X) = \{V_1 | V_2 \rightarrow V_1 \text{ is in } G, V_1 \notin X \text{ and } V_2 \in X\}$ . The spouses of  $X$  is the set  $sp_G(X) = \{V_1 | V_1 \leftrightarrow V_2 \text{ is in } G, V_1 \notin X \text{ and } V_2 \in X\}$ . The adjacents of  $X$  is the set  $adj_G(X) = \{V_1 | V_1 \rightarrow V_2, V_1 \leftarrow V_2, V_1 \leftrightarrow V_2 \text{ or } V_1 - V_2 \text{ is in } G, V_1 \notin X \text{ and } V_2 \in X\}$ . A path from a node  $V_1$  to a node  $V_n$  in  $G$  is a sequence of distinct nodes  $V_1, \dots, V_n$  such that  $V_i \in adj_G(V_{i+1})$  for all  $1 \leq i < n$ . The length of a path is the number of edges in the path. A path is called a cycle if  $V_n = V_1$ . A path is called descending if  $V_i \in pa_G(V_{i+1}) \cup sp_G(V_{i+1})$  for all  $1 \leq i < n$ . The descendants of a set of nodes  $X$  of  $G$  is the set  $de_G(X) = \{V_n | \text{there is a descending path from } V_1 \text{ to } V_n \text{ in } G, V_1 \in X \text{ and } V_n \notin X\}$ .

and  $V_n \notin X\}$ . A path is called strictly descending if  $V_i \in pa_G(V_{i+1})$  for all  $1 \leq i < n$ . The strict descendants of a set of nodes  $X$  of  $G$  is the set  $sde_G(X) = \{V_n | \text{there is a strict descending path from } V_1 \text{ to } V_n \text{ in } G, V_1 \in X \text{ and } V_n \notin X\}$ . The ancestors (resp. strict ancestors) of  $X$  is the set  $an_G(X) = \{V_1 | V_n \in de_G(V_1), V_1 \notin X, V_n \in X\}$  (resp.  $san_G(X) = \{V_1 | V_n \in sde_G(V_1), V_1 \notin X, V_n \in X\}$ ). Note that the definition for strict descendants given here coincides to the definition of descendants given by Richardson (2003). Our definition of descendants is however needed for certain proofs.

A cycle is called a semi-directed cycle if it is descending and  $V_i \rightarrow V_{i+1}$  is in  $G$  for some  $1 \leq i < n$ . A CG is a graph containing only directed and bidirected edges with no semi-directed cycles. An undirected graph is said to be chordal if every cycle of length four or more has an edge between two non-consecutive vertices in the cycle. A CG is said to be connected if there exists a path between every pair of nodes in it. A connectivity component  $C$  of a CG is a maximal set of nodes (wrt to inclusion) st there exists a path between every pair of nodes in  $C$  containing only bidirected edges. The connectivity component of a node  $X$  in a CG  $G$ , denoted  $co_G(X)$ , is the connectivity component in  $G$  to which  $X$  belongs. A subgraph of  $G$  is a subset of nodes and edges in  $G$ . A subgraph of  $G$  induced by a set of its nodes  $X$  is the graph over  $X$  that has all and only the edges in  $G$  whose both ends are in  $X$ .

A node  $C$  is a collider between two nodes  $A$  and  $B$  in a CG  $G$  if there exists edges  $A \rightsquigarrow C \rightsquigarrow B$  in  $G$ . An unshielded collider is a collider where  $A \notin adj_G(B)$  and we then say that  $A$  and  $B$  have an unshielded collider over  $C$ . With a non-collider node  $C$  between two nodes  $A$  and  $B$  we mean that  $A \multimap C \multimap B$  is in  $G$ .

Let  $X$ ,  $Y$  and  $Z$  denote three disjoint subsets of nodes in a CG  $G$ .  $X$  is said to be separated from  $Y$  given  $Z$  iff there exists no path between any node in  $X$  and any node in  $Y$  st: (1) every non-collider on the path is not in  $Z$  and (2) every collider on the path is in  $Z$  or in  $san_G(Z)$ . We denote this by  $X \perp_G Y | Z$ . Likewise, we denote by  $X \perp_p Y | Z$  that  $X$  is independent of  $Y$

given  $Z$  in a probability distribution  $p$ . The independence model induced by  $G$ , denoted as  $I(G)$ , is the set of separation statements  $X \perp_G Y | Z$ .

We say that a probability distribution  $p$  is Markovian with respect to a CG  $G$  when  $X \perp_p Y | Z$  if  $X \perp_G Y | Z$  for all  $X, Y$  and  $Z$  disjoint subsets of  $V$ . We say that  $p$  is faithful to  $G$  when  $X \perp_p Y | Z$  iff  $X \perp_G Y | Z$  for all  $X, Y$  and  $Z$  disjoint subsets of  $V$ . We say that two CGs  $G$  and  $H$  are Markovian equivalent or that they are in the same Markov equivalence class iff  $I(G) = I(H)$ . If  $G$  and  $H$  have the same adjacencies and unshielded colliders, then  $I(G) = I(H)$  (Wermuth and Sadeghi, 2012, Theorem 1).

### 3 Feasible split and merging

In this section we present the feasible split and feasible merge operations. We also show that there exists a set of CGs with a unique minimal set of bidirected edges for each Markov equivalence class and that these CGs can be reached by repeatedly applying feasible splits to any CG in that Markov equivalence class.

---

#### Definition 1. Feasible Split

Let  $C$  denote a connectivity component of  $G$  and  $U$  and  $L$  two disjoint subsets of  $C$  st  $C = U \cup L$  and the subgraph of  $G$  induced by  $U$  is connected. A split of  $C$ , performed by replacing every edge  $X \leftrightarrow Y$  with  $X \rightarrow Y$  st  $X \in U$  and  $Y \in L$ , is feasible iff:

- 1 For all  $A \in sp_G(U) \cap L$ ,  $U \subseteq sp_G(A)$  holds
  - 2 For all  $A \in sp_G(U) \cap L$ ,  $pa_G(U) \subseteq pa_G(A)$  holds
  - 3 For all  $B \in sp_G(L) \cap U$ ,  $sp_G(B) \cap L$  is a complete set
- 

#### Definition 2. Feasible Merging

Let  $U$  and  $L$  denote two connectivity components of  $G$ . A merge between the two components, performed by replacing every edge  $X \rightarrow Y$  with  $X \leftrightarrow Y$  st  $X \in U$  and  $Y \in L$ , is feasible iff:

- 1 For all  $A \in ch_G(U) \cap L$ ,  $pa_G(U) \cup U \subseteq pa_G(A)$  holds
  - 2 For all  $B \in pa_G(L) \cap U$ ,  $ch_G(B) \cap L$  is a complete set
  - 3  $de_G(U) \cap pa_G(L) = \emptyset$
- 

#### Lemma 1. A CG is in the same Markov equivalence class before and after a feasible split.

*Proof.* Let  $G$  be a CG and  $G'$  a graph st  $G'$  is  $G$  with a feasible split performed upon it.  $G$  and

$G'$  are in different Markov equivalence classes or  $G'$  is not a CG iff (1)  $G$  and  $G'$  do not have the same adjacencies, (2)  $G$  and  $G'$  do not have the same unshielded colliders or (3)  $G'$  contains a semi-directed cycle.

First it is clear that the adjacencies are the same before and after the split since the split does not change the adjacencies in  $G$ . Secondly let us assume that  $G$  and  $G'$  do not have the same unshielded colliders. It is clear that a split does not introduce any new unshielded collider, which means that an unshielded collider is removed during the split. Let us say that this unshielded collider is between two nodes  $X$  and  $Y$  over  $Z$  in  $G$  st  $X$  and/or  $Y$  are in  $L$  and  $Z \in U$ . Without loss of generality, let us say that  $X$  is in  $L$ .  $X \leftrightarrow Z$  and  $Y \leftrightarrow Z$  must then hold in  $G$  but  $X \notin adj_G(Y)$ . If  $Y \rightarrow Z$  is in  $G$  this does not fulfill constraint 2 in definition 1, hence  $Y \leftrightarrow Z$  must hold in  $G$ . Now  $Y$  can be either in  $U$  or  $L$ . If  $Y \in U$  constraint 1 in definition 1 is violated and if  $Y \in L$  constraint 3 in definition 1 is violated. Hence we have a contradiction. Finally let us assume a semi-directed cycle is introduced. This can happen iff we have two nodes  $X$  and  $Y$  st  $X \in de_{G'}(Y)$ ,  $X \in U$  and  $Y \in L$ . We know no semi-directed cycle existed in  $G$  before the split. Then  $de_{G'}(Y) \subseteq de_G(Y) \setminus U$  and by definition  $X \notin de_G(Y) \setminus U$ . Hence we have a contradiction.  $\square$

#### Lemma 2. A CG is in the same Markov equivalence class before and after a feasible merging.

*Proof.* Let  $G$  be a CG and  $G'$  a graph st  $G'$  is  $G$  with a feasible merging performed upon it.  $G$  and  $G'$  are in different Markov equivalence classes or  $G'$  is not a CG iff: (1)  $G$  and  $G'$  do not have the same adjacencies, (2)  $G$  and  $G'$  do not have the same unshielded colliders or (3)  $G'$  contains a semi-directed cycle.

First it is clear that the adjacencies are the same before and after the merging since the merging does not change the adjacencies in  $G$ . Secondly let us assume that  $G$  and  $G'$  do not have the same unshielded colliders. It is clear that a merging does not remove any unshielded colliders which means that there has to exist an unshielded collider between two nodes  $X$

and  $Y$  over  $Z$  in  $G'$  that does not exist in  $G$ . It is clear that the following must hold:  $X \notin adj_G(Y)$ ,  $Z \in U$ ,  $X$  and/or  $Y \in L$ . Without loss of generality, let us say that  $X \in L$ , which gives us that  $X \in ch_G(Z)$ . If  $Y \in L$  we have  $Y \in ch_G(Z)$  which contradicts constraint 2 in definition 2. If  $Y \notin L$  we have that  $Y \in pa_G(Z)$  or  $Y \in sp_G(Z)$ , either contradicts constraint 1 in definition 2. Hence an unshielded collider can not be removed. Finally let us assume a semi-directed cycle is introduced. This means that we have three nodes  $X, Y$  and  $Z$  st  $X \in L, Z \in U, Y \notin U \cup L, Z \leftrightarrow X \leftarrow Y$  is in  $G'$  and  $Y \in de_{G'}(Z)$ . However, this violates condition 3 in definition 2, because  $Y \in pa_G(X)$  and  $Y \in de_G(U)$ .  $\square$

We will now show that there exists a set of CGs which have a unique minimal set of bidirected edges for each Markov equivalence class. We also show that this set of edges is shared by all CGs in the class and that the CGs containing no other bidirected edges than the minimal set, can be reached by repeatedly performing feasible splits on any CG in the class.

**Theorem 1.** *For a Markov equivalence class of CGs, there exists a unique minimal (wrt inclusion) set of bidirected edges that is shared by all members of the class.*

*Proof.* Assume to the contrary that there exists two CGs  $G$  and  $G'$  st  $I(G) = I(G')$  and  $G$  and  $G'$  have two different minimal sets of bidirected edges. Now for all ordered pair of nodes  $A$  and  $B$  st  $A \leftrightarrow B$  is in  $G$  but  $A \rightarrow B$  is in  $G'$ , replace  $A \leftrightarrow B$  in  $G$  with  $A \rightarrow B$  and call this new CG  $H$ . Obviously  $H$  has a proper subset of the bidirected edges in  $G$ . We can also see that  $H$  has no semi-directed cycle, because if it had a semi-directed cycle this cycle would also have been in  $G$  or  $G'$  which contradicts that  $G$  and  $G'$  are CGs. Finally we can see that  $I(H) = I(G)$  because  $H$  has the same adjacencies and unshielded colliders as  $G$ . To see the last, note that it is impossible that an unshielded collider  $C \rightarrow A \leftrightarrow B$  is in  $G$  but not in  $H$ , because  $G'$  has no unshielded collider of  $B$  and  $C$  over  $A$  and  $I(G) = I(G')$ .  $\square$

**Theorem 2.** *A CG has the minimal set of bidirected edges for its Markov equivalence class if no feasible split is possible.*

*Proof.* Assume to the contrary there exists a CG  $G$  that does not have the minimal set of bidirected edges for its Markov equivalence class and no split is feasible. Hence there must exist a bidirected edge  $X \leftrightarrow Y$  in  $G$  st  $X \rightarrow Y$  exists in a CG  $G'$  st  $I(G) = I(G')$ . Let  $C$  be the component of  $X$  in  $G$  and  $U$  and  $L$  be two disjoint subsets of  $C$  st  $C = U \cup L$ ,  $X \in U$ ,  $Y \in L$  and the subgraph of  $G$  induced by  $U$  is connected. It is trivial to see that such sets of nodes always must exist. If no split is feasible then we must have that one of the conditions in definition 1 fails for  $U$  and  $L$ . Hence one of the following assumptions must be true.

Assume there exists a node  $A$  st  $A \in sp_G(U) \cap L$  for which  $U \subseteq sp_G(A)$  does not hold. If this is the case there must exist an unshielded collider in  $G$  over a node  $D$  between  $A$  and  $E$  st  $D \in U, D \in sp_G(A), E \in U, E \in sp_G(D), E \notin sp_G(A)$ . This contradicts  $I(G) = I(G')$  since  $D \rightarrow A$  exists in  $G'$ .

Assume there exists a node  $A$  st  $A \in sp_G(U) \cap L$  for which  $pa_G(U) \subseteq pa_G(A)$  does not hold. If this is the case there must exist an unshielded collider in  $G$  over a node  $D$  between  $A$  and  $E$  st  $D \in U, D \in sp_G(A), E \in pa_G(D), E \notin pa_G(A)$ . This contradicts  $I(G) = I(G')$  since  $D \rightarrow A$  exists in  $G'$ .

Assume there exists a  $B \in sp_G(L) \cap U$  for which  $sp_G(B) \cap L$  is not complete. If this is the case there must exist an unshielded collider in  $G$  over a node  $B$  between  $D$  and  $E$  st  $D, E \in L \cap sp_G(B), E \notin adj_G(D)$ . This contradicts  $I(G) = I(G')$  since  $B \rightarrow D$  exists in  $G'$ .

This shows that if one of the constraints in definition 1 fails we can not have  $I(G) = I(G')$  which contradicts the assumption.  $\square$

Finally, it is worth mentioning that we can guarantee that every member of a Markov equivalence class can be reached from any other member of that class by a sequence of feasible splits and mergings. The proof of this result can be seen at

<http://www.ida.liu.se/~jospe/pgm12appendix.pdf>.

This result is not used in this paper but we conjecture that it will play a central role in future work (see section 6).

## 4 Learning algorithm

In this section we present a constraint based algorithm which learns a CG from a probability distribution faithful to some CG. We then prove that the algorithm is correct and that the returned CG contains exactly the minimal set of bidirected edges for its Markov equivalence class.

The algorithm is very similar to the PC algorithm for directed acyclic graphs (Meek, 1995; Spirtes et al., 1993) and shares the same structure with the learning algorithms presented by both Studený for LWF CGs (Studený, 1997) and Peña for AMP CGs (Peña, 2012). The algorithm is shown in algorithm 1 and consists of four separate phases. In phase one (line 1-7) the adjacencies of the CG is recovered. In the second phase (line 8) the unshielded colliders are recovered. Phase three (line 9) then orients some of the remaining edges iff they are oriented in the same direction in all CGs  $G'$  st  $I(G) = I(G')$ . What remains for phase four (line 10-14) is then to orient the rest of the undirected edges st no new unshielded colliders or semi-directed cycles are introduced. This is done according to an algorithm presented in a proof by Koller and Friedman (2009, Theorem 4.13) and is possible since  $H_u$  is chordal as shown in Lemma 8.

The rules used in lines 8-9 in algorithm 1 are shown in figure 1. A rule is said to be applicable if the antecedent is satisfied for an induced subgraph of  $H$ . When a rule is applicable one of the non-arrow edge endings is then replaced with an arrow while the rest of the endings are kept the same. Which edge ending is orientated is shown in the consequent of each rule.

A rule is sound if the orientation introduced by the rule must be shared by every CG which contains the antecedent as an induced subgraph.

**Lemma 3.** *The rules 0 - 3 are sound.*

*Proof.* Rule 0: Since  $B \notin S_{AC}$ ,  $A \in adj_H(B)$  and

## Learning Algorithm

Given a probability distribution  $p$  faithful to an unknown CG  $G$ , the algorithm learns a CG  $H$  st  $I(H) = I(G)$ . Moreover,  $H$  has exactly the minimum set of bidirected edges for its equivalence class.

- 1 Let  $H$  denote the complete undirected graph
- 2 For  $l = 0$  to  $l = |V_H| - 2$
- 3 Repeat while possible
- 4 Select any ordered pair of nodes  $A$  and  $B$  in  $H$  st  $A \in adj_H(B)$  and  $|adj(A) \setminus B| \geq l$
- 5 If there exists  $S \subseteq (adj_H(A) \setminus B)$  st  $|S| = l$  and  $A \perp_p B | S$  then
- 6 Set  $S_{AB} = S_{BA} = S$
- 7 Remove the edge  $A - B$  from  $H$
- 8 Apply rule 0 while possible
- 9 Apply rules 1-3 while possible
- 10 Let  $H_u$  be the subgraph of  $H$  containing only the nodes and the undirected edges in  $H$
- 11 Let  $T$  be the clique tree of  $H_u$
- 12 Order the cliques  $C_1, \dots, C_n$  of  $H_u$  st  $C_1$  is the root of  $T$  and if  $C_i$  is closer to the root than  $C_j$  in  $T$  then  $C_i < C_j$ .
- 13 Order the nodes st if  $A \in C_i$ ,  $B \in C_j$  and  $C_i < C_j$  then  $A < B$
- 14 Orient the undirected edges in  $H$  according to the ordering obtained in line 15
- 15 Return  $H$

**Algorithm 1:** Learning algorithm

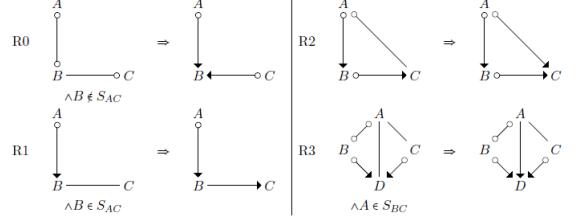


Figure 1: The rules

$C \in adj_H(B)$  but  $A \notin adj_H(C)$  we know that the following configurations can occur:  $A \rightarrow B \leftarrow C$ ,  $A \leftrightarrow B \leftarrow C$ ,  $A \rightarrow B \leftrightarrow C$ ,  $A \leftrightarrow B \leftrightarrow C$ . In any other configuration,  $B$  would be in every separation set of  $A$  and  $C$ . In all these configurations we have  $B \leftrightarrow C$ , so that edge must exist in  $G$ . Rule 1: If the edge was not directed in this direction we would have an unshielded collider of  $A$  and  $C$  over  $B$  that is not in  $G$ , because  $B \in S_{AC}$ . Rule 2: If we did not have the edge oriented in this direction we would have a semi-directed cycle in  $G$ . Rule 3: If the edge was orientated in the opposite direction, by applying rule 2 we would have an unshielded collider of  $B$  and  $C$  over  $A$  that is not in  $G$ , because

$A \in S_{BC}$ . □

**Lemma 4.** *G and H have the same adjacencies after line 7.*

*Proof.* Assume to the contrary that there exists an adjacency between two nodes  $A$  and  $B$  in  $G$  st  $B \notin de_G(A)$  that does not exist in  $H$  or vice versa. We know that, since  $G$  is faithful to  $p$ , all separation statements in  $G$  corresponds to independence statements in  $p$  and vice versa. We will first cover the case where the adjacency is in  $G$  but not in  $H$ . This means that there exists no separation set  $S$  st  $A \perp_p B | S$  since  $A \perp_G B | S$  does not hold for any  $S$ . Hence the prerequisite for line 5 in the algorithm can never be true and so the edge between  $A$  and  $B$  can never be removed and we have a contradiction.

Secondly we will cover if there is an adjacency in  $H$  but not in  $G$ . If there is no adjacency in  $G$  between two nodes  $A$  and  $B$  we know that there has to exist a separation set  $S$  st  $A \perp_p B | S$  and that  $S \subseteq pa_G(A)$  according to the definition of separation. We know, from the paragraph above, that all adjacencies in  $G$  must exist in  $H$  which means that  $pa_G(A) \subseteq adj_H(A)$ . However no such set was found in line 5, hence we have a contradiction. □

**Lemma 5.** *G and H have the same unshielded colliders and adjacencies after line 8.*

*Proof.* From Lemma 4 we know that  $G$  and  $H$  have the same adjacencies. First, assume to the contrary that there exists an unshielded collider in  $G$  but not in  $H$  after line 8. That means we have an unshielded collider between  $A$  and  $B$  over  $C$  st  $C \notin S_{AB}$ ,  $C \in adj_G(A)$  and  $C \in adj_G(B)$  but  $A \notin adj_G(B)$ . According to Lemma 4 the same adjacencies must also hold in  $H$ . We know that if we do not have an unshielded collider in  $H$  we have  $A \rightsquigarrow C \rightarrow B$ . This does however fulfill the prerequisite for rule 0, and hence it should have been applied and we have a contradiction.

Secondly it follows from Lemma 3 that rule 0 is sound so there can be no unshielded colliders in  $H$  that are not in  $G$  after line 8. This brings us to a contradiction. □

**Lemma 6.** *After line 9, H cannot have a subgraph of the form  $A \rightsquigarrow B - C$  without also having the edge  $A \rightarrow C$ .*

*Proof.* Assume the contrary. First we must have  $A \in adj_H(C)$  or rule 1 would orient  $B - C^1$ . We can see that  $H$  can not have the edge  $A \leftrightarrow C$  or rule 2 would be applicable for  $B - C$ . If  $H$  has the edge  $A \rightarrow C$  we are done, which leaves us with  $A - C$ .

Secondly we will study why  $A$  and  $B$  can have an orientation  $A \rightsquigarrow B$ . This can be because of one of four reasons.

Case 1: Edge  $A \rightsquigarrow B$  was orientated using rule 0. This means that there exists a node  $D$  st  $D \rightsquigarrow B$  and  $D \notin adj_H(A)$ .  $D \in adj_H(C)$  must hold or rule 1 would orient  $B - C^1$ , but then rule 3 is applicable for the edge  $B - C$  so this can not be the reason.

Case 2: Edge  $A \rightsquigarrow B$  was orientated using rule 1. This means that the edge  $D \rightsquigarrow A$  st  $D \notin adj(B)$  existed when  $A \rightsquigarrow B$  was oriented.  $D \in adj(C)$  must hold or the edge  $A - C$  would be oriented by rule 1<sup>1</sup>. Restart the proof with  $D \rightsquigarrow A - C$  and it can be seen that this configuration is impossible. Hence this case can not be the reason.

Case 3: Edge  $A \rightsquigarrow B$  was orientated because of rule 3. This means that we have an unshielded collider of two nodes  $D$  and  $E$  st  $D \rightsquigarrow B, E \rightsquigarrow B$   $A \in adj_H(D), A \in adj_H(E)$  and  $D \notin adj_H(E)$ . Now  $D \in adj_H(C)$  and  $E \in adj_H(C)$  must hold or rule 1 would orient  $B - C^1$ . We know that there can be no unshielded collider over  $C$  between  $D$  and  $E$ , otherwise rule 3 would be applicable on  $A - C^1$ . This gives us that we have  $D - C$  and/or  $E - C$ , since if the edge orientated towards  $D$  or  $E$  rule 2 would be applicable. However, with this configuration rule 3 is applicable<sup>1</sup> for edge  $B - C$  so this can not be the reason.

Case 4: Edge  $A \rightsquigarrow B$  was directed using rule 2. This means that the edges  $A \rightsquigarrow D$  and  $D \rightsquigarrow B$  existed in  $H$  when  $A \rightsquigarrow B$  was oriented.  $D \in adj_H(C)$ , otherwise rule 1 would orient the edge  $B - C^1$ .  $D - C$  must hold or rule 2 would cause an orientation of  $B - C$  or  $A - C$ . Restart the proof with  $A \rightsquigarrow D - C$  and it can be seen that this

configuration is impossible, hence case 4 can not be the reason of the orientation.  $\square$

**Lemma 7.** *After line 9,  $H$  can not have a subgraph of the form  $A \leftrightarrow B - C$ .*

*Proof.* Assume the contrary. Then, according to Lemma 6,  $H$  also has the edge  $A \rightarrow C$ . If this is the case then rule 2 is applicable and we have a contradiction.  $\square$

**Lemma 8.** *After line 9, removing all directed edges and bidirected edges from  $H$  results in a chordal graph.*

*Proof.* Assume to the contrary that there exists a non-chordal cycle  $V_1 - V_2 - \dots - V_n - V_1$  and  $n > 3$ . We know that there exists a CG  $G$  that is faithful to the probability distribution  $p$  and has the same unshielded colliders and adjacencies as  $H$  by Lemma 5. From Lemma 3 we also know that the rules are sound, i.e. that the orientations in  $H$  are in  $G$ . Hence we know that there still must exist a valid way to orient the edges in  $H$  that remain undirected after line 9.

Now if we would orient an edge in the cycle st  $V_1 \leftrightarrow V_2 - \dots - V_n - V_1$  it is easy to see that we would have to orient every other edge st  $V_i \rightarrow V_{i+1}$  or we would have a new unshielded collider over some  $V_j$  since  $V_{j-1} \notin \text{adj}_H(V_{j+1})$ . If we would orient all edges in this direction we would however have a semi-directed cycle and hence there exists no possible orientation that results in a CG.  $\square$

**Lemma 9.** *In line 14, when undirected edges are orientated, no new unshielded colliders are introduced to  $H$ .*

*Proof.* It follows directly from Lemma 6 that no undirected edge can be orientated such that it creates an unshielded collider with an edge oriented by rules 0-3. Secondly it is proven (Koller and Friedman, 2009, Theorem 4.13) that no unshiled colliders are created between different undirected edges with the algorithm presented in lines 10-14.  $\square$

<sup>1</sup>Note that if we have  $X \rightsquigarrow Y \rightarrow Z$  in  $H$  and  $X \notin \text{adj}_H(Z)$  we must also have  $Y \in S_{XY}$  or rule 0 would have been applicable in line 8.

**Lemma 10.** *No semi-directed cycle exists in  $H$  after line 15.*

*Proof.* Assume to the contrary, that a semi-directed cycle exists. If the semi-directed cycle is of length 3 then it could have been be created in one of the following ways:

Case 1: All its edges got oriented by rules 0-3. However, this is a contradiction because rule 2 would be applicable and no semi-directed cycle would remain after applying the rule.

Case 2: All its edges got oriented by line 14. However, this is a contradiction by Theorem 4.13 in Koller and Friedman (2009).

Case 3: Some edges got oriented by rules 0-3 and some by line 14. Then, after applying the rules, the cycle contained the configuration  $A \rightarrow B - C$  or  $A \leftrightarrow B - C$  for some nodes  $A, B, C$  in the cycle. The first one is impossible because, otherwise,  $A \rightarrow C$  by Lemma 6 and, thus, there would not be semi-directed cycle of length 3, which contradicts the paragraph above. The latter one is impossible by Lemma 7.

Hence we can have no semi-directed cycle of length 3. Now assume the semi-directed cycle is of length 4. It could have been be created in one of the following ways:

Case 1: All its edges got oriented by rules 0-3. This implies that the semi-directed cycle in  $H$  is actually a cycle with only bidirected edges in  $G$  because the rules are sound. However, this implies that there is an unshielded collider in  $G$  that was not in  $H$ , which contradicts Lemma 5, or that  $H$  had a semi-directed cycle of length 3, which contradicts the paragraph above.

Case 2: All its edges got oriented by line 14. However, this is a contradiction by Theorem 4.13 in Koller and Friedman (2009).

Case 3: Some edges got oriented by rules 0-3 and some by line 14. Then, after applying the rules, the cycle contained the configuration  $A \rightarrow B - C$  or  $A \leftrightarrow B - C$ . The first one is impossible because, otherwise,  $A \rightarrow C$  by Lemma 6 and there would not be semi-directed cycle. The latter one is impossible by Lemma 7.

Hence  $H$  has no semi-directed cycle of length 4. Now, repeating the reasoning for semi-directed cycles of length 5, 6, etc. it is easy to

see that no semi-directed cycle can exist since  $H$  has a bounded number of nodes.

□

**Theorem 3.** *After line 15,  $H$  is a CG and  $I(H) = I(G)$ .*

*Proof.* Lemma 3, 5 and 9 gives that  $I(H) = I(G)$  after line 14. Lemma 10 then refutes that  $H$  can contain a semi-directed cycle. □

**Theorem 4.** *After line 15,  $H$  has exactly the unique minimal set of bidirected edges for its Markov equivalence class.*

*Proof.* It is clear that bidirected edges only can be introduced to  $H$  by rules 0-3. From Lemma 3 it also follows that the rules are sound meaning that all the orientations in  $H$  caused by the rules have to exist in every  $G'$  st  $I(G') = I(G)$ . □

## 5 Conclusion

In this paper we have presented and proved two fundamental elements for the multivariate regression interpretation of CGs. The first element was an algorithm to learn a multivariate regression CG from a probability distribution faithful to some CG. The second element we have presented is a feasible split and a feasible merging st they alter the structure of a CG but do not change the Markov equivalence class of the CG. We have also shown that there exists a set of CGs with a unique minimal set of bidirected edges for each Markov equivalence class and that these CGs can be reached from any CG in the class using the split operation.

## 6 Further Work

Many elements are still not presented or proven for the multivariate regression interpretation of CGs. The natural continuation of the work presented here would be to develop a learning algorithm with weaker assumptions than the one presented. This could for example be a learning algorithm which only assumes that the probability distribution fullfills the composition property. A second natural continuation of the work here would be to use the split and merging operations to prove that Meek's conjecture holds

for the multivariate regression interpretation of CGs, similar way to Peña's work within the LWF interpretation (Peña, 2011). This could then be used to develop learning algorithms that are correct under the composition property.

## Acknowledgments

This work is funded by the Center for Industrial Information Technology (CENIIT) and a so-called career contract at Linköping University, and by the Swedish Research Council (ref. 2010-4808).

## References

- Steen A. Andersson, David Madigan and Michael D. Perlman. 2001. Alternative Markov Properties for Chain Graphs. *Scandinavian Journal of Statistics*, 28:33–85.
- David R. Cox and Nanny Wermuth. 1993. Linear Dependencies Represented by Chain Graphs. *Statistical Science*, 8:204–283.
- David R. Cox and Nanny Wermuth. 1996. *Multivariate Dependencies: Models, Analysis and Interpretation*. Chapman and Hall.
- Mathias Drton. 2009. Discrete Chain Graph Models. *Bernoulli* 15(3):736–753.
- Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models*. MIT Press.
- Steffen L. Lauritzen. 1996. *Graphical Models*. Oxford University Press.
- Zongming Ma, Xianchao Xie and Zhi Geng. 2008. Structural Learning of Chain Graphs via Decomposition. *Journal of Machine Learning Research*, 9:2847–2880.
- Christopher Meek. 1995. Causal Inference and Causal Explanation with Background Knowledge. *Proceedings of 11th Conference on Uncertainty in Artificial Intelligence*, 403–418.
- Jose M. Peña. 2011. Towards Optimal Learning of Chain Graphs. arXiv:1109.5404v1 [stat.ML].
- Jose M. Peña. 2012. Learning AMP Chain Graphs under Faithfulness arXiv:1204.5357v1 [stat.ML].
- Thomas S. Richardson. 2003. Markov Properties for Acyclic Directed Mixed Graphs. *Scandinavian Journal of Statistics*, 30(1):145–157.
- Peter Spirtes, Clark Glymour and Richard Scheines. 1993. *Causation, Prediction, and Search*. Springer-Verlag.
- Milan Studený. 1997. A Recovery Algorithm for Chain Graphs. *International Journal of Approximate Reasoning*, 17:265–293.
- Milan Studený, Alberto Roverato and Šárka Štěpánová. 2009. Two Operations of Merging and Splitting Components in a Chain Graph. *Kybernetika*, 45:208–248.
- Nanny Wermuth and Kayvan Sadeghi. 2012. Sequences of regressions and their independencies. *Invited paper in TEST*. Available online.

# Integer Linear Programming Approach to Learning Bayesian Network Structure: towards the Essential Graph

Milan Studený

Institute of Information Theory and Automation of the ASCR, Czech Republic  
studeny@utia.cas.cz

## Abstract

The basic idea of a geometric approach to learning a Bayesian network (BN) structure is to represent every BN structure by a certain vector. If the vector representative is chosen properly, it allows one to re-formulate the task of finding the global maximum of a score over BN structures as an integer linear programming (ILP) problem. Suitable such a zero-one vector representative is the *characteristic imset*, introduced in (Studený, Hemmecke and Lindner, 2010). In this paper, extensions of characteristic imsets are considered which additionally encode chain graphs without flags equivalent to acyclic directed graphs. The main contribution is the polyhedral description (= in terms of a set of linear inequalities) of the respective domain of the ILP problem. It is just a theoretical result, but it opens the way to the application of ILP software packages in the area of learning a BN structure. The advantage of this approach is that, as a by-product of the ILP optimization procedure, one may get the *essential graph*, which is a traditional graphical BN representative.

## 1 Introduction

Learning *Bayesian network* (BN) structure by a score and search method means to maximize a *quality criterion*  $\mathcal{Q}$ , also named a *score*, which is a real function of the (acyclic directed) graph  $G$  and the observed database  $D$ . The value  $\mathcal{Q}(G, D)$  evaluates how the BN structure defined by the graph  $G$  fits the database  $D$ .

Two important technical assumptions on the criterion  $\mathcal{Q}$  were pinpointed in the literature in connection with computational aspects of this maximization task:  $\mathcal{Q}$  should be *score equivalent* (Bouckaert, 1995) and (additively) *decomposable* (Chickering, 2002).

The geometric approach is to represent every BN structure by a certain vector so that such a criterion  $\mathcal{Q}$  becomes an affine function of the vector representative. This idea was introduced already by Studený (2005) and then deepened in (Studený, Vomlel and Hemmecke, 2010). A suitable (uniquely determined) such a zero-one vector BN representative seems to be the *char-*

*acteristic imset*, introduced at last PGM (Studený, Hemmecke and Lindner, 2010).

Jaakkola et al. (2010) and Cussens (2010; 2011) came independently with an analogous geometric approach. The main difference is that they used certain special zero-one vector codes of (acyclic) directed graphs to represent (non-uniquely) BN structures. On the other hand, they made more progress with the practical use of *integer linear programming* (ILP) tools. To overcome technical problems with the exponential length of their vectors they utilized the idea of reduction of the search space from (de Campos et al., 2009), based on a particular form of databases and criteria occurring in practice.

In (Studený and Haws, 2012), both methods of BN structure vector representation were compared and it was found that the characteristic imset can be viewed as a (many-to-one) linear function of the above mentioned codes of directed graphs. Finally, Lindner (2012) performed some preliminary computational experiments based on the characteristic imset ap-

proach; an overview of this approach has been given in (Studený et al., 2012) and (Hemmecke et al., 2012).

In this paper, an extended vector BN representative is introduced, which encodes both the characteristic imset and a certain special graph (equivalent to an acyclic directed graph). The main result is a *polyhedral characterization* of the domain of the respective ILP problem. More specifically, a set of linear inequalities is presented such that the only vectors with integer components in the polyhedron specified by those inequalities are the above mentioned extended vector representatives.

The inequalities are classified in four groups. The number of inequalities in the first two groups is polynomial in the number of variables (= nodes of the graph), while the number of remaining inequalities is exponential. However, provided the length of the vector representatives is limited/reduced to a polynomial number by the idea of from (de Campos et al., 2009), the number of inequalities in the third group can be reduced to a polynomial number as well. The fourth group of inequalities correspond to acyclicity restrictions. In general, they cannot be reduced to a polynomial number, but because of their natural graphical interpretation, the (modified) cutting plane approach may be applied to solve the respective ILP problems.

Another advantage of this extended vector representative is that one can get, as a result of solving the ILP problem, the *essential graph*, which is known as a standard unique graphical BN representative (Andersson, Madigan and Perlman, 1997).

## 2 Basic concepts

Let  $N$  be a finite non-empty set of *variables*; let's assume  $|N| \geq 2$  to avoid the trivial case. In statistical context, the elements of  $N$  correspond to random variables in consideration; in graphical context, they correspond to nodes.

### 2.1 Graphical concepts

Graphs considered in this paper have  $N$  as the set of nodes and two types of edges between (distinct) nodes  $i, j \in N$ , namely directed edges,

called *arrows*, and denoted like  $i \rightarrow j$  (or  $j \leftarrow i$ ), and undirected edges, called *lines*, and denoted like  $i - j$  (or  $j - i$ ). No multiple edges are allowed between two nodes. If there is an edge between nodes  $i$  and  $j$ , we say they are *adjacent*. A graph is *undirected* if it only has lines; it is *directed* if it only has arrows.

A *cycle* of the length  $m \geq 3$  in a graph  $H$  is a sequence of nodes  $\rho : i_0, i_1, \dots, i_m = i_0$ , where  $i_1, \dots, i_m$  are distinct nodes, and  $i_r$  and  $i_{r+1}$  are adjacent in  $H$  (for each  $r = 0, \dots, m-1$ ). The cycle  $\rho$  is *chordless* if there is no other edge in  $H$  between nodes in  $\{i_1, \dots, i_m = i_0\}$  besides those which form the cycle  $\rho$ . An undirected graph is called *chordal* if it has no chordless cycle of the length  $m \geq 4$ .

The cycle  $\rho$  is *directed* if  $i_r \rightarrow i_{r+1}$  in  $H$  for each  $r = 0, \dots, m-1$ . A directed graph is *acyclic* if it has no directed cycle (of arbitrary length  $m \geq 3$ ). An equivalent definition of an acyclic directed graph  $G$  is that there exists an ordering  $b_1, \dots, b_m$ ,  $m \geq 1$  of all nodes in  $N$  which is consistent with the direction of arrows:  $b_i \rightarrow b_j$  in  $G$  implies  $i < j$ . The set of *parents* of  $i \in N$  in a (directed) graph  $G$  is the set  $pa_G(i) \equiv \{j \in N; j \rightarrow i \text{ in } G\}$ .

The cycle  $\rho$  is *semi-directed* if  $i_0 \rightarrow i_1$  in  $H$  and, for each  $r = 1, \dots, m-1$  one has either  $i_r \rightarrow i_{r+1}$  in  $H$  or  $i_r - i_{r+1}$  in  $H$ . A *chain graph* is a graph without semi-directed cycles.

A set  $C \subseteq N$  is *connected* if every pair of distinct nodes in  $C$  is connected via an undirected path. Maximal connected sets are called *components*. An equivalent definition of a chain graph  $H$  is that there exists an ordering  $C_1, \dots, C_m$ ,  $m \geq 1$  of all its components such that if  $a \rightarrow b$  in  $H$  then  $a \in C_i$  and  $b \in C_j$  with  $i < j$ .

### 2.2 Bayesian network structures

In statistical context, each variable  $i \in N$  is assigned a finite *sample space*  $X_i$  (of possible values); assume  $|X_i| \geq 2$  for each  $i \in N$  to avoid technical problems.

A *Bayesian network* (BN) is a pair  $(G, P)$ , where  $G$  is an acyclic directed graph with the node set  $N$  and  $P$  a *Markovian* probability distribution (with respect to  $G$ ) on the *joint sample space*  $\prod_{i \in N} X_i$ . This means  $P$  satisfies condi-

tional independence restrictions determined by  $G$ ; see (Lauritzen, 1996) for details. The *BN structure* defined by an acyclic directed graph  $G$  is the class of Markovian probability distributions with respect to  $G$  on (fixed)  $\prod_{i \in N} X_i$ .

However, different graph over  $N$  can be *Markov equivalent*, which means they define the same BN structure. Classic graphical characterization of (Markov) equivalent graphs by Verma and Pearl (1991) says they are equivalent iff they have the same adjacencies and the same immoralities. Here, an *immorality* in a graph  $G$  is an induced subgraph (of  $G$ ) for three nodes  $\{a, b, c\}$  in which  $a \rightarrow c \leftarrow b$  (and  $a$  and  $b$  are not adjacent).

*Learning BN structure* means to determine it on the basis of an observed *database*  $D$ , which is a sequence  $x_1, \dots, x_\ell$  of elements of  $\prod_{i \in N} X_i$  ( $\ell \geq 1$  is the length of the database). This is often done by maximizing some *quality criterion* (= score), which is a real function  $\mathcal{Q}$  of two variables: of an acyclic directed graph  $G$  and of a database  $D$ . The value  $\mathcal{Q}(G, D)$  quantitatively evaluates how well the BN structure defined by the graph  $G$  explains the occurrence of the database  $D$ .

Because the aim is to learn a BN structure, a natural requirement is that  $\mathcal{Q}$  should be *score equivalent*, which means that, given any  $D$ ,

$$\mathcal{Q}(G, D) = \mathcal{Q}(H, D)$$

for any pair of Markov equivalent acyclic directed graphs  $G$  and  $H$  over  $N$ .

Additively *decomposable* criterion is a criterion  $\mathcal{Q}$  which can be written as follows:

$$\mathcal{Q}(G, D) = \sum_{i \in N} q_{i|B}(D_{\{i\} \cup pa_G(i)}), \quad (1)$$

where  $q_{i|B}$  for  $i \in N$ ,  $B \subseteq N \setminus \{i\}$  are some real functions and  $D_A$  for  $\emptyset \neq A \subseteq N$  denotes the projection of the database  $D$  to  $\prod_{i \in A} X_i$ . The terms  $q_{i|B}(D_{\{i\} \cup B})$  are named *local scores*.

Well-known quality criteria used in practice are Schwarz's (1978) *Bayesian information criterion* (BIC) and the *Bayesian Dirichlet Equivalence* (BDE) score (Heckerman et al., 1995).

### 2.3 Essential graphs

A kind of standard (unique) graphical representative of a BN structure is the so-called essential graph.

**Definition 1.** Let  $\mathcal{G}$  be a Markov equivalence class of acyclic directed graphs over  $N$ . The *essential graph*  $G^*$  of  $\mathcal{G}$  is defined as follows:

- $a \rightarrow b$  in  $G^*$  if  $a \rightarrow b$  in every  $G$  from  $\mathcal{G}$ ,
- $a - b$  in  $G^*$  if there are graphs  $G_1$  and  $G_2$  in  $\mathcal{G}$  with  $a \rightarrow b$  in  $G_1$  and  $a \leftarrow b$  in  $G_2$ .

This terminology and the first graphical characterization of essential graphs was given by Andersson, Madigan and Perlman (1997). It implies that every essential graph is a chain graph and has no *flag*, by which is meant an induced subgraph for three nodes  $\{a, b, c\}$  in which  $a \rightarrow b - c$  (and  $a$  and  $c$  are not adjacent). One can introduce a graphical concept of equivalence for these graphs, which generalizes Markov equivalence of acyclic directed graphs.

**Definition 2.** Two chain graphs without flags  $G$  and  $H$  over  $N$  are *equivalent* if they have the same adjacencies and immoralities. Given two such graphs, we say that  $H$  is *larger* than  $G$  if, for any  $i, j \in N$ ,  $i \rightarrow j$  in  $H$  implies  $i \rightarrow j$  in  $G$ .

The following characterization of the essential graphs, proved as Corollary 4 in (Studený, 2004), will be utilized later.

**Lemma 1.** Let  $\mathcal{G}$  be an equivalence class of acyclic directed graphs over  $N$  and  $\mathcal{H}$  an equivalence class of chain graphs without flags such that  $\mathcal{G} \subseteq \mathcal{H}$ . Then  $G^*$  is the largest graph in  $\mathcal{H}$ .

## 3 Characteristic imset

This algebraic representative of a BN structure was introduced in (Studený, Hemmecke and Lindner, 2010). For our purpose, the following equivalent definition is suitable.

**Definition 3.** Let  $G$  be an acyclic directed graph over  $N$ . The *characteristic imset* for  $G$  can be introduced as a zero-one vector  $\mathbf{c}_G$  with components  $\mathbf{c}_G(S)$  where  $S \subseteq N$ ,  $|S| \geq 2$ , such that  $\mathbf{c}_G(S) = 1$  iff

$$\exists i \in S \text{ such that } S \setminus \{i\} \subseteq pa_G(i). \quad (2)$$

The point is that two acyclic directed graphs  $G$  and  $H$  over  $N$  are Markov equivalent if and only if  $\mathbf{c}_G = \mathbf{c}_H$ ; see § 3 of (Hemmecke et al., 2012). Moreover, Corollary 2 in (Hemmecke et al., 2012) implies that, for different  $i, j, k \in N$ ,

- (i)  $i$  and  $j$  are adjacent in  $G$  iff  $\mathbf{c}_G(\{i, j\}) = 1$ ,
- (ii)  $i \rightarrow k \leftarrow j$  is an immorality in  $G$  iff  $\mathbf{c}_G(ijk) = 1$  and  $\mathbf{c}_G(ij) = 0$ .

In particular, one can observe that the characteristic imset  $\mathbf{c}_G$  is uniquely determined by its values  $\mathbf{c}_G(S)$  for  $S \subseteq N$ ,  $2 \leq |S| \leq 3$ .

It appears to be suitable to have a formula for the characteristic imset on the basis of any graph  $H$  in the class  $\mathcal{H}$  from Lemma 1. For this purpose one needs the next auxiliary concept.

**Definition 4.** We say that a graph  $H$  over  $S \subseteq N$  has a *super-terminal component* if there exists a non-empty set  $K \subseteq S$  such that

- $K$  is a *complete set* in  $H$ , which means, for each pair of distinct nodes  $i, k \in K$ , one has  $i - k$  in  $H$ ,
- $\forall j \in S \setminus K \quad \forall i \in K$  one has  $j \rightarrow i$  in  $H$ .

It makes no problem see that a super-terminal component  $K$ , if exists, is uniquely determined.

The following result follows directly from Theorem 2 in (Hemmecke et al., 2012):

**Lemma 2.** Let  $H$  be a chain graph without flags equivalent to an acyclic directed graph  $G$ . For any  $S \subseteq N$ ,  $|S| \geq 2$  one has  $\mathbf{c}_G(S) = 1$  iff the induced subgraph of  $H$  for  $S$  (denoted by  $H_S$ ) has a super-terminal component.

### 3.1 Straightforward codes of graphs

Jaakkola et al. (2010) and Cussens (2010; 2011) used a special method for vector encoding (acyclic) directed graphs over  $N$ . The vector  $\eta_G$  encoding  $G$  has components indexed by pairs  $(i|B)$ , where  $i \in N$  and  $B \subseteq N \setminus \{i\}$ . Specifically, it is defined as follows:

$$\eta_G(i|B) = \begin{cases} 1 & B = \text{pa}_G(i), \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

In (Studený and Haws, 2012), the relation of the characteristic imset to this straightforward code of  $G$  was established. Actually,  $\mathbf{c}_G$  is a linear function of  $\eta_G$  given by

$$\mathbf{c}_G(S) = \sum_{i \in S} \sum_{B, S \setminus \{i\} \subseteq B \subseteq N \setminus \{i\}} \eta_G(i|B) \quad (4)$$

for  $S \subseteq N$ ,  $|S| \geq 2$ . Indeed, (4) follows directly from Definition 3: clearly, at most one node  $i \in S$  with  $S \setminus \{i\} \subseteq \text{pa}_G(i)$  exists in an acyclic directed graph  $G_S$ .

### 4 Integer linear programming

The task to maximize a criterion can be reformulated as an *integer linear programming* (ILP) problem. Indeed, by (1) and (3), every decomposable criterion  $\mathcal{Q}$  can be interpreted as a linear function of  $\eta_G$ , where  $G$  falls within the class of acyclic directed graphs over  $N$ .

Jaakkola et al. (2010) gave a finite list of valid inequalities for  $\eta_G$ 's, which characterize them in the sense that the only vectors with integer components satisfying those inequalities are the codes of acyclic directed graphs. Such a domain description, in terms of polyhedral geometry named an *LP relaxation* (of the respective polytope), allows one to turn the learning task into an ILP problem: to optimize a linear function over vectors with integer components within a polyhedron.

Specifically, besides basic non-negativity and equality constraints, Jaakkola et al. (2010) came with the following *cluster inequalities*:

$$1 \leq \sum_{i \in S} \sum_{B \subseteq N \setminus S} \eta_G(i|B) \quad (5)$$

for any  $S \subseteq N$ ,  $|S| \geq 2$ . The meaning of the inequality (5) is that the induced subgraph  $G_S$  has at least one *initial node*, that is, a node  $i \in S$  with no  $j \in S$  with  $j \rightarrow i$  in  $G$ . As  $G_S$  is acyclic, the existence of such a node is obvious.

To overcome the technical problem with the exponential length (in  $|N|$ ) of vectors  $\eta_G$  the idea of *pruning* of their components was applied. The idea taken from (de Campos et al., 2009) is that a particular form of scoring criteria used in practice allows one to conclude (on the

basis of an observed database) that the optimal graph  $G$  has no node  $i \in N$  with large  $|pa_G(i)|$ . Therefore, one can exclude from consideration the respective components of the  $\eta$ -vector. This pruning procedure is time demanding, but useful: as reported in § 6 of (de Campos and Ji, 2011), in practical cases it typically results in the reduction of the parent set cardinality to at most 5, only in a few cases the maximal cardinality was 7 or 8.

To overcome the problem with the exponential number of cluster inequalities Jaakkola et al. (2010) used the method of iterative constraint adding, where they employed the dual formulation (of their approximate LP problems) to guide the choice of a newly added cluster constraint.

Cussens was in (2010) interested in pedigree learning, in which case the parent set cardinality is bounded by 2. However, to ensure the acyclicity of the graph  $G$  he used another trick: the idea of *extending* the vector BN representatives. He added some additional components to the (reduced)  $\eta_G$ -vector which allowed him to encode the total order of nodes consistent with the direction of arrows in the graph  $G$ . Then he introduced easily an LP relaxation for these extended vector representatives. Actually, the number of the added components and the number of inequalities ensuring acyclicity were both polynomial in  $|N|$ .

The other paper by Cussens (2011) was inspired by Jaakkola et al. (2010). Unrestricted BN structure learning was the goal and to overcome the problem with the exponential number of these inequalities Cussens used the *cutting plane* approach.

#### 4.1 ILP with characteristic imsets

Lemma 1 in (Hemmecke et al., 2012) says that every score equivalent and additively decomposable criterion  $\mathcal{Q}$  has the form

$$\begin{aligned} \mathcal{Q}(G, D) &= \mathcal{Q}(G^\emptyset, D) \\ &+ \sum_{S \subseteq N, |S| \geq 2} r_D^\mathcal{Q}(S) \cdot c_G(S), \end{aligned} \quad (6)$$

where  $G^\emptyset$  is the empty graph over  $N$  (= without adjacencies) and  $r_D^\mathcal{Q}$  uniquely determined vec-

tor, depending on the database  $D$  only, called the *revised data vector* (relative to  $\mathcal{Q}$ ). Given  $\mathcal{Q}$ , one usually can derive a mathematical formula for the components of  $r_D^\mathcal{Q}$ . An alternative way, described in (Studený, 2012), is to compute  $r_D^\mathcal{Q}(S)$  for  $S \subseteq N$ ,  $|S| \geq 2$  from local scores.

That means, the criterion  $\mathcal{Q}$  can be interpreted as an affine function (= a linear function plus a constant) of the characteristic imset  $c_G$ , which is a unique BN representative. Thus, to employ the methods of ILP, one has to come with an LP relaxation for characteristic imsets.

In (Studený and Haws, 2012), we transformed the above-mentioned LP relaxation by Jaakkola et al. (2010) through (4) into the framework of characteristic imsets. A pleasant finding was that the cluster inequality (5) takes a neat form

$$\sum_{T \subseteq S, |T| \geq 2} c(T) \cdot (-1)^{|T|} \leq |S| - 1. \quad (7)$$

Another non-trivial observation was that the transformed linear inequalities define an LP relaxation of the characteristic imset polytope. However, since the number of resulting inequalities is super-exponential in  $|N|$ , which is an unpleasant consequence of the many-to-one transformation, this LP relaxation does not seem to be suitable for practical purposes.

Another important fact is that pruning can also be utilized in the context of characteristic imsets. This follows easily from (4): if the components of  $\eta_G$  were pruned to the maximal parent set cardinality  $k$ , then one can assume  $c_G(S) = 0$  for  $S \subseteq N$  with  $|S| > k + 1$  in any optimal graph  $G$ . Then  $r_D^\mathcal{Q}(S)$  for such  $S$  need not be computed.

Lindner (2012) came with another LP relaxation of the characteristic imset polytope. She also used the idea of extending BN vector representatives: the components added to the characteristic imset allowed her to encode acyclic directed graphs defining the characteristic imset. Finally, she performed some preliminary computational experiments based on this approach.

## 5 LP relaxation

In this section we present another LP relaxation for the characteristic imset polytope, also based on the idea of adding components.

### 5.1 Extended vector representative

The result from Lemma 1 motivated the idea to broaden the class of (non-unique) graphical BN representatives to the class of *chain graphs without flags that are equivalent to acyclic directed graphs*. Lemma 2 then motivated the following definition.

**Definition 5.** Let  $H$  be a chain graph over  $N$  without flags (equivalent to an acyclic directed graph). We ascribe to  $H$  a zero-one vector  $(\mathbf{a}_H, \mathbf{c}_H)$  with components given as follows:

$$\mathbf{a}_H(i \rightarrow j) = 1 \iff i \rightarrow j \text{ in } H,$$

where  $i, j \in N$  are distinct, and,

$$\mathbf{c}_H(S) = 1 \iff H_S \text{ has a super-terminal component},$$

for  $S \subseteq N$ ,  $|S| \geq 2$ .

Thus, the  $\mathbf{a}$ -part of the vector encodes the presence of arrows  $i \rightarrow j$  in the graph  $H$  (= codes of *arrowheads*), while the  $\mathbf{c}$ -part is, by Lemma 2, the respective characteristic imset. Note that the number of added components  $|N| \cdot (|N| - 1)$  is polynomial in  $|N|$ .

### 5.2 The list of inequalities

The inequalities are classified in four groups and none of them is superfluous.

The *basic non-negativity inequalities* are:

$$(b.1) \quad \forall i, j \in N \text{ distinct} \quad 0 \leq \mathbf{a}(i \rightarrow j),$$

$$(b.2) \quad \forall S \subseteq N, |S| = 3, 4 \quad 0 \leq \mathbf{c}(S).$$

The *consistency inequalities* mainly relate the  $\mathbf{a}$ -part to the  $\mathbf{c}$ -part: for distinct  $i, j, k \in N$

$$(c.1) \quad \mathbf{a}(i \rightarrow j) + \mathbf{a}(j \rightarrow i) \leq \mathbf{c}(ij),$$

$$(c.2) \quad \mathbf{c}(ij) \leq 1,$$

$$(c.3) \quad 2 \cdot \mathbf{c}(ijk) \leq 2 \cdot \mathbf{c}(ij) + \mathbf{a}(i \rightarrow k) + \mathbf{a}(j \rightarrow k),$$

$$(c.4) \quad \mathbf{a}(i \rightarrow j) + \mathbf{c}(jk) \leq 1 + \mathbf{c}(ijk) + \mathbf{a}(j \rightarrow k),$$

$$(c.5) \quad \begin{aligned} \mathbf{a}(i \rightarrow j) + \mathbf{c}(jk) + \mathbf{c}(ik) \\ \leq 2 + \mathbf{a}(i \rightarrow k) + \mathbf{a}(k \rightarrow j). \end{aligned}$$

The *extension inequalities* ensure that the  $\mathbf{c}$ -part is determined by values  $\mathbf{c}(S)$ ,  $2 \leq |S| \leq 3$ :

$$(e.1) \quad \forall S \subseteq N, |S| \geq 3$$

$$\sum_{i \in S} \mathbf{c}(S \setminus \{i\}) \leq 2 + (|S| - 2) \cdot \mathbf{c}(S),$$

$$(e.2) \quad \forall S \subseteq N, |S| \geq 4$$

$$(|S| - 1) \cdot \mathbf{c}(S) \leq \sum_{i \in S} \mathbf{c}(S \setminus \{i\}).$$

Finally, the *acyclicity inequalities* only concern the  $\mathbf{c}$ -part and ensure that the solution is the graph in the considered class:

$$(a.1) \quad \forall S \subseteq N, |S| \geq 4$$

$$\sum_{T \subseteq S, |T| \geq 2} \mathbf{c}(T) \cdot (-1)^{|T|} \leq (|S| - 1).$$

Observe these are just the transformed cluster inequalities (7). Here is the main result.

**Theorem 1.** Let  $H$  be a chain graph without flags equivalent to an acyclic directed graph over  $N$ . Then the inequalities (b.1)-(b.2), (c.1)-(c.5), (e.1)-(e.2) and (a.1) are valid for the vector  $(\mathbf{a}_H, \mathbf{c}_H)$  from Definition 5. Conversely, if a vector  $(\mathbf{a}, \mathbf{c})$  with integer components satisfies those inequalities, then such (uniquely determined) graph  $H$  exists with  $(\mathbf{a}, \mathbf{c}) = (\mathbf{a}_H, \mathbf{c}_H)$ .

A complete proof is beyond the scope of a conference paper and can be found in (Studený, 2012); in fact, a stronger result is derived there saying that, if (a.1) is omitted, one still gets a code of a certain graph  $H$ , but with possible semi-directed cycles. Theorem 1 also holds with (a.1) replaced by a simplified version, perhaps easier to implement:

$$(a.1^*) \quad \forall S \subseteq N, |S| \geq 4$$

$$\sum_{T \subseteq S, 2 \leq |T| \leq 3} \mathbf{c}(T) \cdot (-1)^{|T|} \leq (|S| - 1).$$

The resulting polyhedron is, however, different. Lindner (2012) mentioned (a.1\*), too.

### 5.3 Interpretation of inequalities

One can give graphical *interpretation* to (most of) the inequalities, on which the proof of their necessity is based.

- (c.1) means that if  $i \rightarrow j$  or  $j \rightarrow i$  is encoded in  $\mathbf{a}$  then  $[i, j]$  is encoded as an edge in  $\mathbf{c}$ ,
- (c.2) means, together with (c.1), that one cannot have simultaneously  $i \rightarrow j$  and  $j \rightarrow i$  in  $H$ ,
- (c.3) allows one to conclude that if  $\mathbf{c}(ijk) = 1$  then  $H_{ijk}$  has a super-terminal component,
- (c.4) prevents  $H$  to have a flag  $i \rightarrow j - k$ ,
- (c.5) says that  $H$  has not a semi-directed 3-cycle of the form  $i, j, k, i$  with  $i \rightarrow j$ ,
- (e.1) means: if  $S$  has at least 3 subsets  $T$  with  $|T| = |S| - 1$  with  $\mathbf{c}(T) = 1$  then  $\mathbf{c}(S) = 1$ ,
- (e.2) if  $\mathbf{c}(S) = 1$  then at least  $|S| - 1$  sets  $T \subset S$  with  $|T| = |S| - 1$  and  $\mathbf{c}(T) = 1$  exists.
- (a.1) means, loosely said, that the graph  $H_S$  has at least one initial node; it forbids the existence of a chordal semi-directed/undirected cycle composed just of the nodes of  $S$ .

#### 5.4 The idea of the sufficiency proof

First, one observes that the above inequalities, together with the assumption that  $(\mathbf{a}, \mathbf{c})$  has integer components, imply that  $0 \leq \mathbf{c}(S) \leq 1$  for any  $S \subseteq N$ ,  $|S| \geq 2$ . Given such a vector  $(\mathbf{a}, \mathbf{c})$ , define a graph  $H$ :

$$\begin{aligned} i \rightarrow j \text{ in } H &\Leftrightarrow \mathbf{a}(i \rightarrow j) = 1, \\ i - j \text{ in } H &\Leftrightarrow \mathbf{c}(ij) = 1 \& \\ &\mathbf{a}(i \rightarrow j) = 0 \& \mathbf{a}(j \rightarrow i) = 0. \end{aligned}$$

Then the inequalities allow one to show that  $H$  is 3-acyclic (= has no semi-directed cycle of the length 3) and has no flags. The next step is to show that, for distinct  $i, j, k \in N$ ,  $\mathbf{c}(ijk) = 1$  iff  $H_{ijk}$  has a super-terminal component. Using (e.1)-(e.2), this observation is extended to any  $S \subseteq N$ ,  $|S| \geq 2$  in place of  $ijk$ . Finally, (a.1) is used to show that  $H$  has no semi-directed or undirected chordless cycle of the length  $m \geq 4$ .

#### 6 Summary of the whole procedure

A pre-processing step should be the *pruning* in the context of characteristic imsets; see § 4.1. The result should be a cache of values  $r_D^Q(S)$  for

$S \in \mathcal{T}$ , where  $\mathcal{T} \subseteq \{S \subseteq N; |S| \geq 2\}$  is a class of sets closed under subsets such that  $\mathbf{c}_G(S) = 0$  for  $S \notin \mathcal{T}$  and any optimal  $G$ . The hope is that  $\mathcal{T}$  will consist of sets of small cardinality.

The first ILP problem to be solved is to maximize the function

$$(\mathbf{a}, \mathbf{c}) \mapsto \sum_{S \in \mathcal{T}} r_D^Q(S) \cdot \mathbf{c}(S)$$

over the domain of vectors with integral components specified by the inequalities from § 5.2. The number of consistency inequalities is polynomial in  $|N|$  and, provided  $|\mathcal{T}|$  is small, a small number of extension inequalities is applicable.

The number of acyclicity inequalities cannot be reduced to a polynomial amount, but one can apply the idea of *iterative constraint adding*. In the first iteration, the acyclicity inequalities are omitted. The solution of the respective ILP problem corresponds to a graph  $H$  with possible semi-directed/undirected chordless cycles. One can identify minimal sets  $S$  of nodes in  $H$  forming those cycles. The corresponding acyclicity inequalities are incorporated in the current list of inequalities and a revised ILP problem is solved. This procedure is repeated until either a solution without those cycles is found or memory overfill.

Once such a graph  $H$  is found, the respective *essential graph*  $G^*$  can be obtained as follows. The idea is to fix the  $\mathbf{c}$ -part of the vector solution and formulate the second ILP problem: to minimize the objective

$$(\mathbf{a}, \mathbf{c}) \mapsto \sum_{i, j \in N, i \neq j} \mathbf{a}(i \rightarrow j)$$

instead, under non-negativity and consistency constraints. By Lemma 1, the solution should correspond to the essential graph  $G^*$  for  $\mathbf{c}$ .

#### Conclusions

The main theoretical advantage of the presented approach (in comparison with the other ILP approaches) is that the solution is a unique BN representative and the vector representatives are more compressed. The augmentation does not kill this comparative advantage because the extended characteristic imsets are still

more than  $\frac{|N|}{3}$ -times shorter than the straightforward graph codes. The number of inequalities is also smaller than Lindner (2012) used.

Another fine feature is that the presented approach allows one to get directly the *essential graph* as a result of solving the ILP problem, that is, to avoid in this way the need for additional graph-reconstruction procedure.

Nevertheless, the theoretical assumptions (or ambitions) must be tested in practice. Realize that there is also strong influence of the pruning on the efficiency of the overall procedure and it is not clear at present whether a smaller number of inequalities is better than a tighter LP relaxation involving more inequalities. Thus, the proposed LP relaxation should become a basis for computational experiments, made in cooperation with foreign colleagues.

## Acknowledgments

This research has been supported by the grant GAČR n. 201/08/0539.

## 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.
- James Cussens. 2010. Maximum likelihood pedigree reconstruction using integer programming. In *Workshop on Constraint Based Methods for Bioinformatics*, pages 9–19.
- James Cussens. 2011. Bayesian network learning with cutting planes. In *27th Conference on Uncertainty in Artificial Intelligence*, pages 153–160.
- Cassio P. de Campos, Zhi Zeng and Qiang Ji. 2009. Structure learning Bayesian networks using constraints. In *26th International Conference on Machine Learning*, pages 113–120.
- Cassio P. de Campos and Qiang Ji. 2011. Efficient structure learning Bayesian networks using constraints. *Journal of Machine Learning Research*, 12:663–689.
- David Heckerman, Dan Geiger and David M. Chickering. 1995. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 20:194–243.
- Raymond Hemmecke, Silvia Lindner and Milan Studený. 2012. Characteristic imsets for learning Bayesian network structure. To appear in *International Journal of Approximate Reasoning*, see doi:10.1016/j.ijar.2012.04.001.
- Tommi Jaakkola, David Sontag, Amir Globerson and Marina Meila. 2010. Learning Bayesian network structure using LP relaxations. In *JMLR Workshop and Conference Proceedings, volume 9: AISTATS*, pages 358–365.
- Steffen L. Lauritzen. 1996. *Graphical Models*, Clarendon Press.
- Silvia Lindner. 2012. Discrete optimization in machine learning - learning Bayesian network structures and conditional independence implication. PhD thesis, TU Munich.
- 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 of 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, Jiří Vomlel and Raymond Hemmecke. 2010. A geometric view on learning Bayesian network structures. *International Journal of Approximate Reasoning*, 51(5):578–586.
- Milan Studený, Raymond Hemmecke and Silvia Lindner. 2010. Characteristic imset: a simple algebraic representative of a Bayesian network structure. In *5th European Workshop on Probabilistic Graphical Models*, pages 257–264.
- Milan Studený and David Haws. 2012. On polyhedral approximations of polytopes for learning Bayesian networks. Submitted to *Journal of Algebraic Statistics*; previous working paper available on <http://arxiv.org/abs/1107.4708>.
- Milan Studený, David Haws, Raymond Hemmecke and Silvia Lindner. 2012. Polyhedral approach to statistical learning graphical models. In *2nd CREST-SBM International Conference*, World Scientific, pages 346–372.
- Milan Studený. 2012. LP relaxations and pruning for characteristic imsets. Research report n. 2323, Institute of Information Theory and Automation of the ASCR; available through [staff.utia.cas.cz/studeny/f18.html](http://staff.utia.cas.cz/studeny/f18.html).
- Thomas Verma and Judea Pearl. 1991. Equivalence and synthesis of causal models. In *6th Conference on Uncertainty in Artificial Intelligence*, pages 220–227.

# The Bayesian Chow-Liu Algorithm

Joe Suzuki

Osaka University, Japan

suzuki@math.sci.osaka-u.ac.jp

## Abstract

Given data, not knowing the distribution, we wish to construct a forest (Markov graph) relative to which the description length is minimized, connecting edges with larger estimated mutual information of each pair of random variables step by step (the Chow-Liu algorithm) to balance simplicity of the forest and fitness of the data, where the random variables are not to be either discrete or continuous. To this end, we construct a Bayesian measure over the data sequences, and propose the Bayesian estimator of mutual information. The Bayesian measure is partially from (Ryabko 2009), but our version can deal with any random variable even if no density function exists. We show that the estimator is consistent, and it is considered to be more robust than the existing approaches because it does not estimate one specific histogram from data. Numerical experiments demonstrate that the proposed method works efficiently enough to deal with practical problems.

## 1 Introduction

In many applications of statistical machine learning such as data mining and pattern recognition, we often need to capture the dependencies among random variables from data. The obtained relation can be usually expressed by graphical models such as Markov networks and Bayesian networks (Pearl 1988). However, as the number of random variables increases, it is hard to obtain the exact estimation because its computation increases exponentially.

In this paper, we restrict the distribution of random variables expressed by a Markov graph to a limited form of distributions expressed by a tree (we identify the random variables with the vertexes in the graph). If the distribution is known, such an approximation can be executed via the Chow-Liu algorithm (Chow and Liu, 1968) which continues to connect a pair of vertexes with the largest mutual information if the connection does not make any loop (otherwise, the pair will not be considered for an edge in the future) until no candidate exists. Although the search is done in a top down manner, it is guaranteed that the resulting tree expresses a distribution such that the K-L divergence from

the true distribution is minimized (Section 2.1).

In our problem, only the data is available while the true distribution is not known. Given  $n$  examples consisting of attribute values, a naive way to construct a distribution expressing a tree is to maximum likelihood estimate the values of mutual information based the examples (Section 2.2). On the other hand, (Suzuki, 1993) considered to minimize the description length rather than the K-L divergence by estimating each mutual information in a Bayesian manner: construct measures  $R_X^n, R_Y^n, R_{XY}^n$  over  $\mathcal{X}^n, \mathcal{Y}^n, \mathcal{X}^n \times \mathcal{Y}^n$ , where  $\mathcal{X}, \mathcal{Y}$  are the ranges of random variables  $X, Y$ . Then, the Bayesian estimator is expressed by  $\frac{1}{n} \log \frac{R_{XY}^n(x^n, y^n)}{R_X^n(x^n)R_Y^n(y^n)}$  for given examples  $(x^n, y^n) \in \mathcal{X}^n \times \mathcal{Y}^n$ , reflecting simplicity of each forest as well as the likelihood of the examples to the tree (Section 2.3).

The main purpose of this paper is to extend the Chow-Liu algorithm so that it can deal with arbitrary random variables: the existing methods deal with only random variables taking values in finite sets. Suppose that all the variables are continuous and a simultaneous density function exists. Then, constructing a kernel function to which the training data fit may show better

performance in some cases. However, in reality, in any data base, some attributes are discrete, and others are continuous. So, assuming only continuous variables would be too restrictive for the multivariate case. We do not assume that the random variables are either discrete or continuous. To this end, we need to extend the notion of description length and Bayesian estimators of mutual information. We require such a Bayesian estimator to be (strongly) consistent, i.e., the estimator should converge to the true one as  $n \rightarrow \infty$  with probability one.

There are many ways to estimate mutual information. Most conventional approaches were to quantize the ranges  $\mathcal{X}, \mathcal{Y}$  for estimation and to increase the number of bins in the histogram as the sample size  $n$  grows: (Darbellay and Vajda 1999) considered to update the bins adaptively based on the samples obtained thus far (strong consistency was not proved for the method); (Wang et al 2005) applied a similar idea to estimation of Kullback-Leibler divergence; and recently (Silva and Narayanan 2010) obtained a consistent estimator of mutual information using a similar but more general principle.

We construct a measure  $g^n$  over  $\mathcal{X}^n$  that is universal in the sense  $\frac{1}{n} \log \frac{f_X^n(x^n)}{g_X^n(x^n)}$  diminishes with probability as one  $n \rightarrow \infty$  for any  $f_X^n$ . The idea is to prepare a nested sequence  $\{A_k\}$  of histograms and to estimate the density function  $f_k^n(x^n)$  by  $g_k^n(x^n)$  for each histogram  $A_k$ , assuming that the density function  $f_X$  exists for the random variable  $X$ . Then, we mix them with weights  $\{w_k\}$  such that  $\sum_{k=1}^{\infty} w_k = 1, w_k > 0$  to obtain the value  $g_X^n(x^n) = \sum_{k=1}^{\infty} w_k g_k^n(x^n)$ . For the measure, (Ryabko 2009) proved universality for any  $f_X$  such that  $h(f_k) \rightarrow h(f_X)$  as  $k \rightarrow \infty$  (Section 3.2).

We extend the universal measure  $g^n$  so that the random variables can be either discrete or continuous. The basic idea is to replace the Lebesgue measure  $\lambda$  with another supporting measure  $\eta$  if no density function exists for the random variable  $X$ . In particular, if  $\mathcal{X}$  is finite, the  $g_X^n$  reduces to  $R_X^n$  by choosing an appropriate  $\eta$ , as shown in Section 3.3.

The proposed extended Bayesian estimator is expressed by  $\frac{1}{n} \log \frac{g_{XY}^n(x^n, y^n)}{g_X^n(x^n)g_Y^n(y^n)}$ , where  $g_X^n, g_Y^n, g_{XY}^n$  are universal density functions with respect to  $\eta \neq \lambda$ . We show that the estimator actually converges to the mutual information with probability one as  $n \rightarrow \infty$  (Section 4.1).

There are many consistent estimators of mutual information. The proposed Bayesian estimator has several merit over them. For example, by maximizing it for each step, the associate description length assuming the graph  $(V, E)$  to be a forest will be minimized among  $E$  (Section 4.2).

In the last section (Section 5), we list other merits of the Bayesian estimator and state future works.

## 2 The Chow-Liu Algorithm

### 2.1 The Original Version

Let  $V$  be a finite set, and  $E$  a subset of  $\mathcal{E} := \{\{i, j\} \subseteq V | i \neq j\}$ . In this paper, we say such a pair  $(V, E)$  to be a *graph*. The sequences  $(i_0, \dots, i_m), (i_m, \dots, i_0) \in V^{m+1}$  are said to be a pair of *paths* connecting  $\{i_0, i_m\}$  of length  $m$  in  $(V, E)$  if  $i_0, \dots, i_m$  are different and  $\{i_0, i_1\}, \dots, \{i_{m-1}, i_m\} \in E$ . We say the graph  $(V, E)$  is a *forest* if a pair of paths connecting each element in  $\mathcal{E}$  is unique (if it exists), and a *tree* if there exists a unique path connecting each element in  $\mathcal{E}$ .

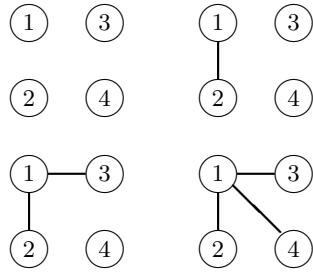
In this paper, we consider the following algorithm for graphs: given  $\{w_{i,j}\}_{\{i,j\} \in \mathcal{E}}$  such that  $w_{i,j} \geq 0, w_{i,j} = w_{j,i}$ , it outputs a tree maximizing  $\sum_{\{i,j\} \in E} w_{i,j}$  among graphs  $(V, E)$  that express trees (Kruskal's algorithm, Aho 1974):

1.  $\mathcal{E} \leftarrow \{\{i, j\} | i, j \in V, i \neq j\}$
2.  $E \leftarrow \{\}$
3. while( $\mathcal{E} \neq \phi$ ) for  $\{i, j\} \in \mathcal{E}$  maximizing  $w_{i,j}$ 
  - (a)  $\mathcal{E} \leftarrow \mathcal{E} \setminus \{\{i, j\}\}$
  - (b)  $(V, E \cup \{\{i, j\}\})$  is a forest  $\implies E \leftarrow E \cup \{\{i, j\}\}$

If the initial  $\mathcal{E}$  is replaced by  $\{\{i, j\} | i, j \in V, i \neq j, w_{i,j} > 0\}$ , then the resulting graph is a forest rather than a tree (the generalized Kruskal algorithm).

**Example 1.** Suppose that the values of  $\{w_{i,j}\}_{\{i,j\} \in \mathcal{E}}$  are given in the table below. The largest value in the table is twelve for  $(i, j) = (1, 2)$ , so we connect them first. The second largest is ten for  $(i, j) = (1, 3)$ , so we connect them. The third largest is eight for  $(i, j) = (2, 3)$ , but connecting them makes a loop, so we do not connect them. The fourth largest is six for  $(i, j) = (1, 4)$ , so we connect them. But, we cannot connect any further for the rest  $(i, j) = (2, 4), (3, 4)$ .

|           |    |    |   |   |   |   |
|-----------|----|----|---|---|---|---|
| $i$       | 1  | 1  | 2 | 1 | 2 | 3 |
| $j$       | 2  | 3  | 3 | 4 | 4 | 4 |
| $w_{i,j}$ | 12 | 10 | 8 | 6 | 4 | 2 |



Let  $X^{(1)}, \dots, X^{(N)}$  be random variables that take values in finite sets  $\mathcal{X}^{(1)}, \dots, \mathcal{X}^{(N)}$ , respectively. Let  $P_{1,\dots,N}(x^{(1)}, \dots, x^{(N)})$ ,  $P_i(x^{(i)})$ , and  $P_{i,j}(x^{(i)}, x^{(j)})$  be the probabilities of  $(X^{(1)}, \dots, X^{(N)}) = (x^{(1)}, \dots, x^{(N)}) \in \mathcal{X}^{(1)} \times \dots \times \mathcal{X}^{(N)}$ ,  $X^{(i)} = x^{(i)} \in \mathcal{X}^{(i)}$ , and  $(X^{(i)}, X^{(j)}) = (x^{(i)}, x^{(j)}) \in \mathcal{X}^{(i)} \times \mathcal{X}^{(j)}$ , respectively. Also, let  $H(i)$ ,  $I(i, j)$ , and  $H(1, \dots, N)$  be the entropy of  $X^{(i)}$ , the mutual information of  $\{X^{(i)}, X^{(j)}\}$ , and the simultaneous entropy of  $\{X^{(1)}, \dots, X^{(N)}\}$ , respectively.

Let  $V := \{1, \dots, N\}$ , and  $E \subseteq \mathcal{E} = \{\{i, j\} \subseteq V | i \neq j\}$ . Assuming  $(V, E)$  is a tree, we identify  $V$  with  $\{X^{(1)}, \dots, X^{(N)}\}$  to approximate  $P_{1,\dots,N}(x^{(1)}, \dots, x^{(N)})$  by

$$Q_{1,\dots,N}(x^{(1)}, \dots, x^{(N)}) = \frac{\prod_{\{i,j\} \in E} P_{i,j}(x^{(i)}, x^{(j)})}{\prod_{i \in V} P_i(x^{(i)})^{d_i - 1}}$$

(Dendroid distribution), where  $d_i := |\{j \in$

$$V | \{i, j\} \in E\}|^1.$$

Suppose that we rearrange the indexes  $1, \dots, N$  so that  $i \leq j$  if path  $(1, \dots, i, \dots, j)$  exists. Then, for each  $j = 2, \dots, N$ , the  $i$  such that  $i < j$  and  $\{i, j\} \in E$  is unique:

$$Q_{1,\dots,N}(x^{(1)}, \dots, x^{(N)}) \\ = P_{1}(x^{(1)}) \prod_{\{i,j\} \in E, i < j} \frac{P_{i,j}(x^{(i)}, x^{(j)})}{P_i(x^{(i)})}.$$

In 1968, Chow and Liu showed that if we apply Kruskal's algorithm with weights  $w_{i,j} := I(i, j)$ , then, the resulting tree minimizes the Kullback-Leibler divergence

$$D(P_{1,\dots,N} || Q_{1,\dots,N}) \\ = \sum_{i \in V} H(i) - H(1, \dots, N) - \sum_{\{i,j\} \in E} I(i, j)$$

depends only on  $E$  in the last term.

## 2.2 The Chow-Liu Algorithm based on ML Estimation

Suppose that  $x^n := \{(x_i^{(1)}, \dots, x_i^{(N)})\}_{i=1}^n \in (\prod_{j=1}^N \mathcal{X}^{(j)})^n$  have been emitted i.i.d. by  $P_{1,\dots,N}$ . Let  $c_{1,\dots,N}(x^{(1)}, \dots, x^{(N)})$ ,  $c_{i,j}(x^{(i)}, x^{(j)})$ , and  $c_i(x^{(i)})$  be the numbers of occurrences of  $(X^{(1)}, \dots, X^{(N)}) = (x^{(1)}, \dots, x^{(N)}) \in (\mathcal{X}^{(1)} \times \dots \times \mathcal{X}^{(N)})$ ,  $(X^{(i)}, X^{(j)}) = (x^{(i)}, x^{(j)}) \in \mathcal{X}^{(i)} \times \mathcal{X}^{(j)}$ , and  $X^{(i)} = x^{(i)} \in \mathcal{X}^{(i)}$ , in  $x^n$ , respectively. If we divide them by  $n$ , we obtain the values of  $\hat{P}_{1,\dots,N}(x^{(1)}, \dots, x^{(N)})$ ,  $\hat{P}_{i,j}(x^{(i)}, x^{(j)})$ , and  $\hat{P}_i(x^{(i)})$ , respectively.

If we define  $\hat{Q}_{1,\dots,N}$ ,  $\hat{H}(i)$ ,  $\hat{H}(1, \dots, N)$ ,  $\hat{I}(i, j)$  in terms of  $\hat{P}_{1,\dots,N}$ ,  $\hat{P}_{i,j}$ ,  $\hat{P}_i$  (the maximum likelihood estimators), we obtain a tree minimizing

$$D(\hat{P}_{1,\dots,N} || \hat{Q}_{1,\dots,N}) \\ = \sum_{i \in V} \hat{H}(i) - \hat{H}(1, \dots, n) - \sum_{\{i,j\} \in E} \hat{I}(i, j)$$

## 2.3 The Chow-Liu Algorithm based on the MDL

Let  $R^n(i)$  and  $R^n(i, j)$  be measures over  $(\mathcal{X}^{(i)})^n$  such that  $\sum R^n(i) \leq 1$  and over  $(\mathcal{X}^{(i)} \times \mathcal{X}^{(j)})^n$

<sup>1</sup>|S| denotes the cardinality of set  $S$ .

such that  $\sum R^n(i, j) \leq 1$ , respectively. Then, if we define the measure over  $(\mathcal{X}^{(1)} \times \cdots \times \mathcal{X}^{(N)})^n$  by

$$\begin{aligned} R^n(1, \dots, N|E) &:= \frac{\prod_{\{i,j\} \in E} R^n(i, j)}{\prod_{i \in V} R^n(i)^{d_i-1}} \\ &= \prod_{\{i,j\} \in E} \frac{R^n(i, j)}{R^n(i)R^n(j)} \prod_{i \in V} R^n(i), \end{aligned}$$

then the Chow-Liu algorithm maximizing the Bayesian estimator

$$J(i, j) := \frac{1}{n} \log \frac{R^n(i, j)}{R^n(i)R^n(j)} \quad (1)$$

in each step minimizes the associated description length (Rissanen 1978)

$$\begin{aligned} L(x^n|E) &:= -\log R^n(1, \dots, N|E) \\ &= -\sum_{i \in V} \log R^n(i) - \sum_{\{i,j\} \in E} \log \frac{R^n(i, j)}{R^n(i)R^n(j)}. \end{aligned}$$

However, the  $E$  minimizing the description length  $L(x^n|E)$  may not converge to the true  $E$  as  $n \rightarrow \infty$ . In fact, for example, if the values of  $R^n(i)$  and  $R^n(i, j)$  are uniform and do not depend on  $x^n$ , the quantity  $L(x^n|E)$  does not give any information to estimate  $E$ , even if we properly specify the prior probability  $P(E)$  over the subsets of  $\mathcal{E}$ .

Hereafter, for simplicity, we assume that the prior probability  $P(E)$  is uniform, so that, given  $x^n$ , we evaluate  $(V, E)$  only by  $L(x^n|E)$  rather than by  $-\log P(E) + L(x^n|E)$ .

On the other hand, if we apply the Krichevsky-Trofimov estimator (Krichevsky and Trofimov, 1981) such as

$$R^n(i) := \frac{\Gamma(n + \alpha^{(i)}a)\Gamma(a)^{\alpha^{(i)}}}{\Gamma(\alpha^{(i)}a) \prod_{x^{(i)} \in \mathcal{X}^{(i)}} \Gamma(c_i[x^{(i)}] + a)} \quad (2)$$

$$\begin{aligned} R^n(i, j) \\ &:= \{\Gamma(n + \alpha^{(i)}\alpha^{(j)}a)\Gamma(a)^{\alpha^{(i)}\alpha^{(j)}}\}/\{\Gamma(\alpha^{(i)}\alpha^{(j)}a) \\ &\quad \prod_{x^{(i)} \in \mathcal{X}^{(i)}, x^{(j)} \in \mathcal{X}^{(j)}} \Gamma(c_{i,j}[x^{(i)}, x^{(j)}] + a)\}, \end{aligned}$$

with parameter  $a = 1/2$ , we obtain<sup>2</sup>

$$\begin{aligned} -\log R^n(i) &\approx n\hat{H}(i) + \frac{\alpha^{(i)} - 1}{2} \log n, \\ -\log R^n(i, j) &\approx n\hat{H}(i, j) + \frac{\alpha^{(i)}\alpha^{(j)} - 1}{2} \log n, \\ \log \frac{R^n(i, j)}{R^n(i)R^n(j)} &\approx n\hat{I}(i, j) - \frac{(\alpha^{(i)} - 1)(\alpha^{(j)} - 1)}{2} \log n, \end{aligned}$$

and thus

$$\begin{aligned} L(x^n|E) &\approx n \sum_{i \in V} \{\hat{H}(i) + \frac{\alpha^{(i)} - 1}{2n} \log n\} \\ &\quad - n \sum_{\{i,j\} \in E} \{\hat{I}(i, j) - \frac{1}{2n}(\alpha^{(i)} - 1)(\alpha^{(j)} - 1) \log n\}. \end{aligned}$$

Since  $\frac{1}{n} \log R^n(i) \rightarrow H(i)$  and  $\frac{1}{n} \log R^n(i, j) \rightarrow H(i, j)$ , with probability one as  $n \rightarrow \infty$ , we find that  $J(i, j)$  is a consistent estimator of  $I(i, j)$ :

$$\begin{aligned} J(i, j) &\approx \hat{I}(i, j) - \frac{1}{2n}(\alpha^{(i)} - 1)(\alpha^{(j)} - 1) \log n \\ &\rightarrow I(i, j), \end{aligned} \quad (3)$$

so that the  $E$  minimizing  $L(x^n|E)$  converges to the true  $E$  as  $n \rightarrow \infty$  with probability one.

Notice that the resulting  $w_{i,j} := J(i, j)$  could be negative, and the resulting graph  $(V, E)$  is a forest rather than a tree if we apply the generalized Kruskal algorithm. J. Suzuki (1993) proposed to apply  $J(i, j)$  in (3) rather than  $\hat{I}(i, j)$  to compare in each step to balance simplicity of the  $(V, E)$  and fitness of the  $x^n$  to  $(V, E)$ . On the other hand, the method based on ML estimation considers only fitness to the  $x^n$ , and eventually over-fitting occurs although the ML estimator is consistent for large  $n$ .

### 3 Universal Measure and Description Length

#### 3.1 Universal Coding for Finite Sources

Suppose that a sequence of random variables  $\{X_i\}_{i=1}^n$  are emitted i.i.d. by probability

<sup>2</sup> $a_n \approx b_n$  denotes  $a_n - b_n$  converges to a constant as  $n \rightarrow \infty$ .

$p^n(x^n) = \prod_{i=1}^n p(x_i)$  for  $x^n = (x_1, \dots, x_n)$  with entropy  $H(p) := \sum_{x \in A} -p(x) \log p(x)$ , where

$A$  is a finite set in which each  $X_i$  takes values. Then, there exists  $q^n$  (see (2)) such that  $\sum_{x^n \in A^n} q^n(x^n) \leq 1$ , and  $-\frac{1}{n} \log q^n(x^n) \rightarrow H(p)$  for any  $p$  with probability one as  $n \rightarrow \infty$ . For example,

$$q^n(x^n) := \frac{\Gamma(\frac{m}{2}) \prod_{a \in A} \Gamma(c_n[a] + \frac{1}{2})}{\Gamma(n + \frac{m}{2}) \Gamma(\frac{1}{2})^m},$$

with  $m := |A|$  (cardinality of  $A$ ) satisfies such a property (Cover 1995), where  $c_n[a]$  is the number of occurrences of  $a \in A$  in  $x^n \in A^n$ , and  $\Gamma$  is the gamma function. We also notice from the Shannon-McMillan-Breiman theorem that  $-\frac{1}{n} \log p^n(x^n) \rightarrow H(p)$  for any  $p$ , which can be also obtained by the strong law of large numbers:

$$-\frac{1}{n} \log p^n(x^n) = \frac{1}{n} \sum_{i=1}^n -\log p(x_i)$$

( $\{-\log p(X_i)\}_{i=1}^n$  are independent random variables). Thus, we have

**Proposition 1.** There exists  $q^n$  such that  $\sum_{x^n \in A^n} q^n(x^n) \leq 1$ , and  $\frac{1}{n} \log \frac{p^n(x^n)}{q^n(x^n)} \rightarrow 0$  for any  $p$  with probability one as  $n \rightarrow \infty$ .

Hereafter, we denote  $L(x^n) := -\log q^n(x^n)$ .

### 3.2 Estimation of Density Functions

Suppose that a sequence of random variables  $\{X_i\}_{i=1}^n$  are emitted i.i.d. by density function  $f^n(x^n) := \prod_{i=1}^n f(x_i)$  for  $x^n = (x_1, \dots, x_n)$ . Let  $\mathcal{X}$  be a range in which each  $X_i$  takes values. We construct a sequence  $\{A_k\}_{k=0}^\infty$  such that  $A_0 := \{\mathcal{X}\}$  and  $A_{k+1}$  is a refinement of  $A_k$ .

**Example 2.** If  $\mathcal{X} = [0, 1)$ , then  $A_0 = \{[0, 1)\}$ , and the sequence

$$\begin{aligned} A_1 &= \{[0, 1/2), [1/2, 1)\} \\ A_2 &= \{[0, 1/4), [1/4, 1/2), [1/2, 3/4), [3/4, 1)\} \\ &\dots \\ A_k &= \{[0, 2^{-(k-1)}), [2^{-(k-1)}, 2 \cdot 2^{-(k-1)}), \\ &\dots, [(2^{k-1} - 1)2^{-(k-1)}, 1)\} \\ &\dots \end{aligned}$$

satisfies the condition.

For each  $k$ , we define projection  $s_k : \mathcal{X}^n \rightarrow A_k^n$  by  $x^n \mapsto a^n$  if  $x^n \in a^n \in A_k^n$ . We denote by  $\lambda$  the Lebesgue measure of  $\mathbb{R}$  with  $\lambda^n(a^n) = \prod_{i=1}^n \lambda(a_i)$  for  $a^n = (a_1, \dots, a_n)$ , and by  $p_k^n(a^n) := \prod_{i=1}^n p_k(a_i)$  the probability of  $s_k(X^n) = a^n \in A_k^n$ .

Since  $s_k(X^n)$  is i.i.d., there exists  $q_k^n$  (Proposition 1) such that

$$\frac{1}{n} \log \frac{p_k^n(s_k(x^n))}{q_k^n(s_j(x^n))} \rightarrow 0$$

for any  $p_k$ . If we define  $g_k^n(x^n) := \frac{q_k^n(s_k(x^n))}{\lambda^n(s_k(x^n))}$ , we construct a measure over  $\mathcal{X}^n$  with  $\{\omega_k\}_{k=1}^\infty$  such that  $\sum \omega_k = 1$ ,  $\omega_k > 0$  to define  $g^n(x^n) := \sum_{k=1}^\infty \omega_k g_k^n(x^n)$ . Notice  $\int_{x^n \in \mathcal{X}^n} g^n(x^n) dx^n = 1$ .

Let  $f_k$  be the density function associated with  $A_k$ , and  $h(f)$  the differential entropy of density function  $f$ .

**Proposition 2** (Ryabko 2009). Fix  $\{A_k\}_{j=1}^\infty$ . Then, for arbitrary  $f$  such that  $h(f_k) \rightarrow h(f)$  as  $j \rightarrow \infty$

$$\frac{1}{n} \log \frac{f^n(x^n)}{g^n(x^n)} \rightarrow 0.$$

### 3.3 Generalization

**Lemma 1** (The Radon-Nikodym Theorem (Billingsley, 1995)). For  $\sigma$ -finite measures<sup>3</sup>  $\mu, \nu$  on the measure space with entire set  $\Omega$  and  $\sigma$ -set field  $\mathcal{F}$ , the following conditions ( $\mu$  is absolutely continuous with respect to  $\nu$ ) are equivalent:

1. there exists  $f$  such that  $\mu(D) = \int_D f(x) d\nu(x)$  for  $D \in \mathcal{F}$
2.  $\mu \ll \nu$ , i.e.  $\nu(D) = 0 \implies \mu(D) = 0$  for  $D \in \mathcal{F}$

<sup>3</sup>A measure  $\nu$  is  $\sigma$ -finite if there exists  $\{A_i\}_{i=1}^\infty$  such that  $\nu(A_i) < \infty$  and  $\cup A_i = \Omega$ .

We denote such an  $f$  (*Radon-Nikodym derivative* (Billingsley, 1995) of  $\mu$  with respect to  $\eta$ ) by  $\frac{d\mu}{d\nu}$ .

In Section 3.2, we assumed  $\mu \ll \lambda$  for an unknown probability measure  $\mu$ , and construct  $g = \frac{d\nu}{d\lambda}$  such that  $\nu \ll \lambda$ .

Now we consider the general case  $\mu \not\ll \lambda$ . Choose  $\eta$  such that  $\mu \ll \eta \neq \lambda$  to estimate the density function  $\frac{d\mu}{d\eta}$  with respect to  $\eta$  instead.

Then, estimation of  $\frac{d\mu}{d\eta}$  is similar except that  $\lambda, \lambda^n$  are replaced by  $\eta, \eta^n$ .

Hereafter, we assume the existence of a underlying supporting  $\sigma$ -finite measure  $\eta$  such that  $\mu \ll \eta$ , so that  $f = \frac{d\mu}{d\eta}$  and  $g = \frac{d\mu}{d\eta}$ .

**Example 3.** Suppose  $B = \{1, 2, \dots\}$ , and that the following sequence is given:

$$\begin{aligned} B_0 &= \{\{1, 2, \dots\}\} \\ B_1 &= \{\{1\}, \{2, \dots\}\} \\ B_2 &= \{\{1\}, \{2\}, \{3, \dots\}\} \\ &\dots \\ B_l &= \{\{1\}, \dots, \{l\}, \{l+1, \dots\}\} \end{aligned}$$

For each  $l$ , we define projection  $t_l : \mathcal{Y}^n \rightarrow B_l^n$  by  $y^n \mapsto b^n$  if  $y^n \in b^n \in B_l^n$ . Then  $m = l + 1$ , and if the source is i.i.d.,

$$\nu_l^n(t_l(y^n)) := \frac{\Gamma(\frac{m}{2}) \prod_{b \in B_l} \Gamma(c_n[b] + \frac{1}{2})}{\Gamma(n + \frac{m}{2}) \Gamma(\frac{1}{2})^m},$$

where  $c_n[b]$  is the number of occurrences of  $b \in B_l$  in  $s_l(y^n) \in B_l^n$ . If we choose  $\eta$  as  $\eta(\{j\}) := \frac{1}{j(j+1)}$  for  $j = 1, 2, \dots$ , thus

$$\eta(\{l+1, \dots\}) = \sum_{j=l+1}^{\infty} = \frac{1}{l+1},$$

then we have  $\mu \ll \eta$ . Then, we can compute

$$\frac{d\nu_l^n}{d\eta^n} = \frac{\nu_l(t_l(y^n))}{\prod_{i=1}^n \eta(t_l(y_i))}$$

to obtain  $\frac{d\nu^n}{d\eta^n} = \sum_{l=1}^{\infty} \omega_l \frac{d\nu_l^n}{d\eta^n}$ . The obtained measure  $\nu^n$  is asymptotically close to  $\mu^n$  in the sense

of Theorem 1 as  $n \rightarrow \infty$ .

#### 4 Estimation of Mutual Information

Let  $X, Y$  be random variables with ranges  $\mathcal{X}, \mathcal{Y}$ . We assume that the measures  $\mu_X, \mu_Y$  of  $X, Y$  are absolutely continuous with respect to  $\sigma$ -finite measures  $\eta_X, \eta_Y$ , respectively. By  $\eta_X \otimes \eta_Y$  we denote the product measure of  $\eta_X, \eta_Y$ , i.e.  $\eta_X \otimes \eta_Y(dx, dy) = \eta_X(dx)\eta_Y(dy)$ .

Choose  $\{A_k\}, \{B_l\}$  so that the differential entropy of the density functions  $f_{X,k}, f_{Y,l}, f_{XY,k,l}$  over  $A_k, B_l, A_k \times B_l$  converge to

$$f_X = \frac{d\mu_X}{d\eta_X}, f_Y = \frac{d\mu_Y}{d\eta_Y}, f_{XY} = \frac{d\mu_{XY}}{d(\eta_X \otimes \eta_Y)}$$

as  $k, l \rightarrow \infty$ . Then, there exist  $g_X^n, g_Y^n, g_{XY}^n$  such that with probability one as  $n \rightarrow \infty$   $\frac{1}{n} \log \frac{f_X^n(x^n)}{g_X^n(x^n)} \rightarrow 0$ ,  $\frac{1}{n} \log \frac{f_Y^n(y^n)}{g_Y^n(y^n)} \rightarrow 0$ , and  $\frac{1}{n} \log \frac{f_{XY}^n(x^n, y^n)}{g_{XY}^n(x^n, y^n)} \rightarrow 0$ . Thus,

$$\begin{aligned} &\frac{1}{n} \log \frac{g_{XY}^n(x^n, y^n)}{g_X^n(x^n)g_Y^n(y^n)} \\ &- \frac{1}{n} \log \frac{f_{XY}^n(x^n, y^n)}{f_X^n(x^n)f_Y^n(y^n)} \rightarrow 0. \end{aligned}$$

From the Shannon-McMillan-Breiman theorem,

$$\begin{aligned} &\frac{1}{n} \log \frac{f_{XY}^n(x^n, y^n)}{f_X^n(x^n)f_Y^n(y^n)} \\ &= \frac{1}{n} \sum_{i=1}^n \log \frac{f_{XY}(x_i, y_i)}{f_X(x_i)f_Y(y_i)} \rightarrow I(X, Y) \end{aligned}$$

which can be also obtained from the strong law of large numbers. Thus,

**Theorem 1.** Given  $x^n \in \mathcal{X}^n, y^n \in \mathcal{Y}^n$ ,

$$\frac{1}{n} \log \frac{g_{XY}^n(x^n, y^n)}{g_X^n(x^n)g_Y^n(y^n)}$$

is a consistent estimator of mutual information  $I(X, Y)$ .

**Example 4.** We construct  $\nu_{k,l}^n(s_k(x^n), t_l(y^n))$  as

$$\frac{\Gamma(\frac{m}{2}) \prod_{a \in A_k} \prod_{b \in B_l} \Gamma(c_n[a, b] + \frac{1}{2})}{\Gamma(n + \frac{m}{2}) \Gamma(\frac{1}{2})^m},$$

based on the sequence  $A_k, B_l$ ,  $k, l = 0, 1, 2, \dots$ , where  $m = 2^k(l+1)$ , and  $\{A_k\}$  and  $\{B_l\}$  have been constructed in Examples 2 and 3, respectively, and  $c_n[a, b]$  is the number of occurrences of  $a \in A_k$  and  $b \in B_l$  in  $(s_k(x^n), t_l(y^n)) \in A_k^n \times B_l^n$ . Then, we can calculate

$$\frac{d\nu_{k,l}^n}{d\eta^n} = \frac{\nu_{k,l}(s_k(x^n), t_l(y^n))}{\prod_{i=1}^n \lambda(s_k(x_i)) \prod_{i=1}^n \eta(t_l(y_i))}$$

to obtain  $\frac{d\nu^n}{d\eta^n} = \sum_{k,l} \omega_{k,l} \frac{d\nu_{k,l}^n}{d\eta^n}$ .

## 5 A Generalized Version of the Chow-Liu Algorithm based on the MDL

Let  $g^n(i) := g_X^n(x^n)$  with  $X = X^{(i)}$  and a supporting measure  $\eta_i$ , and  $g^n(i, j) := g_{XY}^n(x^n, y^n)$  with  $X = X^{(i)}$ ,  $Y = X^{(j)}$  and the supporting measure  $\eta_i \otimes \eta_j$ . We define the Bayesian measure over  $\mathcal{X}^n$  relative to  $(V, E)$ , where  $\mathcal{X} := \mathcal{X}^{(1)} \times \dots \times \mathcal{X}^{(N)}$

$$\begin{aligned} g^n(1, \dots, N|E) &:= \frac{\prod_{\{i,j\} \in E} g^n(i, j)}{\prod_{i \in V} g^n(i)^{d_i-1}} \\ &= \prod_{\{i,j\} \in E} \frac{g^n(i, j)}{g^n(i)g^n(j)} \prod_{i \in V} g^n(i), \end{aligned}$$

the description length with respect to  $(V, E)$  and  $\eta := \eta_1 \otimes \dots \otimes \eta_N$

$$\begin{aligned} L_\eta(x^n|E) &:= -\log g^n(1, \dots, N|E) \\ &= -\sum_{i \in V} \log g^n(i) - \sum_{\{i,j\} \in E} \log \frac{g^n(i, j)}{g^n(i)g^n(j)}, \end{aligned}$$

and the Bayesian estimator

$$J(i, j) := \frac{1}{n} \log \frac{g^n(i, j)}{g^n(i)g^n(j)}.$$

From Theorem 1, we can choose as  $J(i, j)$  a consistent estimator of  $I(i, j)$ .

**Example 5.** Let  $\{A_k\}, \{B_l\}, \{A_k \times B_l\}$  be the sequences constructed in Examples 2,3 and 4. In order to obtain the score  $g^n(i)$ , we consider  $(\{A_k\}, \lambda)$ . Given  $x^n = (x_1, \dots, x_n) \in A^n$ , we obtain the score  $g^n(i)$  for the weights  $\{w_k\}_{k=1}^K$  and functions  $s$  and  $\lambda$ . For  $k = 1, \dots, K$ , the

$s$  returns  $a \in A_k$  such that  $x_j \in a$  for each  $x_j$ , and the  $\lambda$  returns the measure of  $a \in A_k$ , where  $|A_k|$  denotes the cardinality of  $A_k$ .

For each  $k = 0, 1, \dots, K$ , the following algorithm returns  $g_k^n(x^n)$  given  $x^n$  and  $A_k$ , and we obtain  $g^n(i) := \sum_l w_k g_k^n(x^n)$  given  $\{w_k\}$ .

1.  $c[a] := 0$  for  $a \in A_k$ ;
2.  $g_k^n := 1$ ;
3. for  $h = 1, \dots, n$ 
  - (a)  $a := s(x_j)$ ;
  - (b)  $c[a] := c[a] + 1$ ;
  - (c)  $g_k^n := g_k^n * \frac{c[a] + 1/2}{h + |A_k|/2} / \lambda(a)$ ;

It is easy to find that the computation in the proposed algorithm is linear with number  $n$  of the examples. So, if the computation is large, then that will be due to the choice of  $\{A_k\}_{k=1}^K$ .

If we apply binary search for step 3 (a), then  $\log_2 |A_k|$  comparisons are required. So, at most  $O(L \log |A_k|)$  computation is required for the  $K$  cycles. Some might think that  $|A_K|$  would be eventually large. But in reality, we cannot make  $|A_k|$  so large unless  $n$  is fairly large. In fact, if  $|A_k|$  is too large, then each  $c[a]$  will be small, so that the value of  $g_k^n$  is not significant compared with  $\{g_r^n\}_{r < k}$ , and does not affect the resulting value of so much.

Similarly, for each  $k = 0, 1, \dots, K$  and  $l = 0, 1, \dots, L$  the following algorithm returns  $g_{kl}^n(x^n)$  given  $x^n$  and  $A_k, B_l$ , and we obtain  $g^n(i, j) := \sum_{k,l} w_{kl} g_{kl}^n(x^n)$  given  $\{w_{kl}\}$ .

1.  $c[a, b] := 0$  for  $a \in A_k, b \in B_l$ ;
2.  $g_{kl}^n := 1$ ;
3. for  $h = 1, \dots, n$ 
  - (a)  $a := s(x_j); b := t(y_j)$ ;
  - (b)  $c[a, b] := c[a, b] + 1$ ;
  - (c)  $g_{kl}^n := g_{kl}^n * \frac{c[a, b] + 1/2}{h + |A_k||B_l|/2} / (\lambda(a)\eta(b))$ ;

Table 1 shows the values  $J(i, j)$  and its computation time for  $I(i, j) = 2.0$  (The logarithm base is two) and  $K = 8, 64$  and  $n = 100, 1000$ .

| $(K, L)$ | $n$  | $J(i, j)$ | time (ms) |
|----------|------|-----------|-----------|
| (2,4)    | 100  | 0.0612    | 1.23      |
| (2,4)    | 1000 | 0.0531    | 10.67     |
| (4,8)    | 100  | 0.0428    | 1.69      |
| (4,8)    | 1000 | 0.0342    | 14.71     |

Table 1: The value  $J(i, j)$  and its averaged computation time

We generate  $(x^n, y^n) \in (A \times B)^n$  one hundred times to obtain the arithmetic average of  $J(i, j)$ . The data  $(x^n, y^n)$  are generated so that they are independent ( $I(X, Y) = 0$ ). We find that if  $K$  is too small, we only obtain an approximation even when  $n$  is large, and that for large  $K$ , large  $n$  is required for convergence.

In the current problem, for any method, the computation is eventually high, but this is due to the nature of the problem, not due to the proposed method.

## 6 Concluding Remarks

In this paper, we proposed the Bayesian estimator of mutual information which has several merits:

1. consistent: the true mutual information is obtained as  $n$  grows;
2. Bayesian: maximizing the sum of mutual information over the chosen edges leads to minimizing the description length relative to the forest for each  $n$ ;
3. nonparametric: no assumption about any specific parameters is required;
4. robust: compared to existing approaches (Wang et al, 2005)(Silva and Narayan 2010) because the proposed approach does not seek any one histogram but evaluates mutual information based on mixed values of those candidate histograms; and
5. general: applicable to any random variable.

Future works includes figuring out more applications of the universal Bayesian measure. Thus far, it has been found that a similar

approach can be applied to Bayesian network structure estimation when both discrete and continuous variables are present (Suzuki 2011).

## References

- P. Billingsley. *Probability & Measure* (1995): (3rd ed.). New York : Wiley.
- C. K. Chow and C. N. Liu. "Approximating discrete probability distributions with dependence trees". *IEEE Transactions on Information Theory*, IT-14(3):462–467, May (1968).
- T. M. Cover and J. A. Thomas. *Elements of Information Theory* (1995): (2nd ed.). New York : Wiley. *Elements of information theory*: John Wiley & Sons, New York, NY, (1991).
- G. A. Darbellay and I. Vajda. "Estimation of the information by an adaptive partitioning of the observation space". *IEEE Trans. Information Theory*, 45(4):1315-1321, 5 (1999).
- R.E. Krichevsky and V.K. Trofimov, 'The Performance of Universal Encoding", *IEEE Trans. Information Theory*, Vol. IT-27, No. 2, pp. 199-207 (1981)
- Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*, Morgan-Kaufmann (1988)
- J.Rissanen, "Modeling by shortest data description". *Automatica* 14: 465-471 (1978).
- B. Ryabko. "Compression-Based Methods for Non-parametric Prediction and Estimation of Some Characteristics of Time Series." *IEEE Trans. on Inform. Theory*, 55(9):4309-4315 (2009).
- Jorge Silva and Shrikanth Narayanan, Non-product data-dependent partitions for mutual information estimation: Strong consistency and applications, *IEEE Transactions on Signal Processing* (2010).
- J. Suzuki "A Construction of Bayesian Networks from Databases on the MDL principle", *Uncertainty in Artificial Intelligence*, Washington DC, July 1993.
- J. Suzuki, "The Universal Measure for General Sources and its Application to MDL/Bayesian Criteria", Data Compression Conference 2011, Snowbird, Utah, 2011, page 478 (one page abstract).
- Q. Wang, S. Kulkarni, and S. Verd'u. "Divergence estimation of continuous distributions based on data-dependent partitions." *IEEE Trans. Information Theory*, 51(9):3064-3074, 9 (2005).

# Learning high-dimensional mixed graphical models with missing values

Inma Tur and Robert Castelo

Universitat Pompeu Fabra, Barcelona Spain

{inma.tur, robert.castelo}@upf.edu

## Abstract

Current biomedical instrumentation enables monitoring an increasing amount of mixed discrete and continuous variables. This results in multivariate samples amenable for their analysis using mixed graphical models. The dimension of these data sets, with a number of variables  $p$  much larger than the number of observations  $n$ , precludes the direct application of classical learning algorithms and specific procedures that work under the  $p \gg n$  setting are required for that purpose. Yet, a further obstacle to the wide application of these procedures in the biomedical field arises from the fact that missing observations often occur in clinical and genotype data. The high-dimension of  $p$  impedes approaching the problem by simple complete-case analysis and increases substantially the computational burden if we want to use multiply-imputed data sets. Here we show that using limited-order correlations to learn mixed graphical models from data with  $p \gg n$  enables a straightforward and effective application of complete-case analysis to the missing data problem. More importantly, because complete-case analysis is only appropriate under the restrictive assumption that data are missing completely at random, we adapt an expectation-maximization algorithm to the limited-order correlation framework and demonstrate its better suitability under the less stringent assumption of data being missing at random.

## 1 Introduction

Mixed graphical Markov models (GMM) are statistical models representing distributions of discrete and continuous random variables (r.v.) that share a subset of conditional independence restrictions represented by a marked undirected graph  $G$  (Lauritzen and Wermuth, 1989). These models constitute a powerful tool to analyze the increasing amount of data recorded by the ever evolving technologies in the biomedical field, which lead to multivariate data sets of  $p$  r.v. and  $n$  observations where usually  $p \gg n$ .

When the data set is complete, learning the structure of the graph underlying the data is a well-studied problem. In the classical framework where  $n \gg p$ , most relevant aspects can be found in the books of Lauritzen (1996) and Edwards (2000). In the opposite setting, where  $p \gg n$ , Edwards et al. (2010) restrict the learning problem to decomposable mixed GMMs

whereas Tur and Castelo (2011) provide an approach without that restriction.

An important problem in the biomedical field is that missing values arise often in clinical and genotype data due to intrinsic properties of biomarkers and molecules that preclude their recording. A straightforward solution, popularly known as complete-case analysis, is to omit observations with missing values. However, the high dimension of  $p$  increases the likelihood of having at least one missing value at each observation, and therefore, too few complete multivariate samples may be left to directly apply a graphical model learning algorithm.

Imputation approaches, such as in (Broman et al., 2003) for genotype data from experimental crosses, enable also using existing learning algorithms for complete data. A major challenge when directly applying this approach is that to preserve the sampling properties of the

data, multiply-imputed data sets are required and this can substantially increase the computational cost of learning a graphical model.

In this paper we show that using limited-order correlations in the  $p \gg n$  setting (Tur and Castelo, 2011) permits a straightforward and effective application of complete-case analysis. This strategy, however, is only appropriate under the restrictive assumption that data is missing completely at random (MCAR), and it leads to biased estimates of the parameters otherwise. Moreover, even when the MCAR assumption is reasonable, complete-case analysis can yield estimates of higher variability and a loss of statistical power (Little and Rubin, 2002).

For this reason, we extend the learning procedure of Tur and Castelo (2011) to employ an expectation-maximization (EM) algorithm that works under the less stringent assumption of data missing at random (MAR). We show that this provides more satisfying results in both, the MAR and MCAR scenarios.

In Sections 2 and 3 we give an overview of the necessary theory of mixed GMMs and the so-called non-rejection rate, the quantity we employ to learn these graphical models from data with  $p \gg n$ . In Section 4 we describe the EM algorithm in the context of the specific statistical test employed to calculate the non-rejection rate. In Section 5 we show experimental results under different missing data scenarios. Finally, a short discussion is provided in Section 6.

## 2 Mixed Graphical Markov Models

Mixed GMMs represent distributions involving discrete r.v.  $I_\delta$ ,  $\delta \in \Delta$ , and continuous r.v.  $Y_\gamma$ ,  $\gamma \in \Gamma$ , such that an undirected marked graph  $G = (V, E)$  is defined with  $p$  marked vertices  $V = \Delta \cup \Gamma$  and edge set  $E \subseteq V \times V$ . We assume that the joint distribution of the variables  $X = (I, Y)$  is conditional Gaussian (CG-distribution), that is,  $Y \sim \mathcal{N}_{|\Gamma|}(\mu(i), \Sigma(i))$  for each joint discrete level  $i \in \mathcal{I}$ . Considering that discrete genotypes affect mean gene expression values only, we restrict the learning problem to homogeneous mixed GMMs where  $\Sigma(i) \equiv \Sigma$  is constant through  $i \in \mathcal{I}$ .

### 2.1 Decomposable Mixed GMMs

An important subclass of mixed GMMs is defined by decomposable marked graphs.

**Definition 1.** A triple  $(A, B, C)$  of disjoint subsets of  $V$  forms a decomposition of an undirected marked graph  $G$  if  $V = A \cup B \cup C$  and the following three conditions hold: 1.  $C$  is a complete subset of  $V$ ; 2.  $C$  separates  $A$  from  $B$ ; and 3.  $C \subseteq \Delta$  or  $B \subseteq \Gamma$ .

Thus, an undirected marked graph  $G$  is decomposable if there exists a proper decomposition  $(A, B, C)$  such that the subgraphs  $G_{A \cup C}$  and  $G_{B \cup C}$  are decomposable. When this holds, we say that  $C_1 = \{A \cup C\}$  and  $C_2 = \{B \cup C\}$  are cliques and,  $S = \{C\}$  is a separator of  $G$ .

### 2.2 Maximum Likelihood Estimates of Mixed GMMs

Let  $\mathcal{X} = \{x^{(\nu)}\} = \{(i^{(\nu)}, y^{(\nu)})\}$  be a sample of  $\nu = 1, \dots, n$  i.i.d. observations from a homogeneous CG-distribution. For an arbitrary subset  $A \subseteq V$ , we abbreviate to  $i_A = i_{A \cap \Delta}$ ,  $\mathcal{I}_A = \mathcal{I}_{A \cap \Delta}$  and  $y_A = y_{A \cap \Gamma}$  and the following sampling statistics are defined:

$$n(i) = \#\left\{\nu : i^{(\nu)} = i\right\}, \quad (1)$$

$$s(i) = \sum_{\nu: i^{(\nu)}=i} y^{(\nu)}, \quad (2)$$

$$\bar{y}(i) = s(i)/n(i), \quad (3)$$

$$ss(i) = \sum_{\nu: i^{(\nu)}=i} y^{(\nu)}(y^{(\nu)})^T, \quad (4)$$

$$ssd(i) = ss(i) - s(i)s(i)^T/n(i), \quad (5)$$

$$ssd(A) = \sum_{i_A \in \mathcal{I}_A} ssd(i_A), \quad (6)$$

$$ssd_A(A) = \sum_{i_A \in \mathcal{I}_A} ssd_{A \cap \Gamma}(i_A), \quad (7)$$

$$ssd = ssd(V) \text{ and } ssd_A = ssd_A(V). \quad (8)$$

Lauritzen (1996, Prop. 6.10) shows that if  $n \geq |\Gamma| + |\mathcal{I}|$  then the likelihood function attains its maximum almost surely if and only if  $n(i) > 0$  for all  $i \in \mathcal{I}$ . Then, the maximum likelihood estimate (MLE) is given as having moment characteristics equal to the empirical moments, i.e.,

$$\hat{p}(i) = n(i)/n, \quad \hat{\mu}(i) = \bar{y}(i), \quad \hat{\Sigma} = ssd/n. \quad (9)$$

Analogously, decomposable mixed GMMs admit explicit MLEs: in the homogeneous case, (Lauritzen, 1996, Prop. 6.21) shows that the MLE exists almost surely if and only if  $n(i_C) \geq |C \cap \Gamma| + |\mathcal{I}_C|$  for all cliques  $C$  of  $G$  and  $i_C \in \mathcal{I}_C$ . In that case, it is given with the following canonical parameters (Lauritzen, 1996, pg. 189):

$$\hat{p}(i) = \prod_{j=1}^k \frac{n(i_{C_j})}{n(i_{S_j})}, \quad (10)$$

$$\hat{h}(i) = n \left\{ \sum_{j=1}^k [ssd_{C_j}(C_j)^{-1} \bar{y}_{C_j}(i_{C_j})]^{|\Gamma|} - [ssd_{S_j}(S_j)^{-1} \bar{y}_{S_j}(i_{S_j})]^{|\Gamma|} \right\}, \quad (11)$$

$$\hat{K} = n \left\{ \sum_{j=1}^k [ssd_{C_j}(C_j)^{-1}]^{|\Gamma|} - [ssd_{S_j}(S_j)^{-1}]^{|\Gamma|} \right\}, \quad (12)$$

where  $S_1 = \emptyset$  and in Eq.(12)  $[M]^{|\Gamma|}$  is a  $|\Gamma| \times |\Gamma|$  matrix obtained from a  $|A| \times |A|$  matrix  $M = \{m_{\gamma\eta}\}_{|A| \times |A|}$  with  $A \subseteq \Gamma$  such that  $[M]_{\gamma\eta}^{|\Gamma|} = m_{\gamma\eta}$  if  $\gamma, \eta \in A$  and  $[M]_{\gamma\eta}^{|\Gamma|} = 0$  otherwise. Analogously, in Eq. (11)  $[M]^{|\Gamma|}$  is a  $|\Gamma|$ -length vector obtained from a  $|A|$ -length vector  $M$ .

### 3 The Non-rejection Rate

In order to address the problem of learning mixed GMMs from data with  $p \gg n$  and missing values we will use the limited-order correlation approach introduced by Tur and Castelo (2011) and based on the *non-rejection rate* (Castelo and Roverato, 2006). This approach requires testing conditional independencies  $X_\alpha \perp\!\!\!\perp X_\beta | X_Q$  using a likelihood ratio test between two decomposable models.

Without loss of generality, assume that  $V = \{\alpha, \beta, Q\}$  with  $V = \Delta \cup \Gamma$ . Let  $(\gamma, \eta)$  denote a pair of continuous variables ( $\gamma, \eta \in \Gamma$ ), and  $(\delta, \gamma)$  a pair of mixed variables ( $\delta \in \Delta, \gamma \in \Gamma$ ), so that either  $Q = V \setminus \{\gamma, \eta\}$  or  $Q = V \setminus \{\delta, \gamma\}$  as the conditioning subset. In the pure continuous case, the likelihood ratio statistic raised to the power  $2/n$  corresponding to  $\gamma \perp\!\!\!\perp \eta | Q$  is (Lauritzen, 1996, pg. 192):

$$\Lambda_{\gamma\eta.Q} = \frac{|ssd_\Gamma||ssd_{\Gamma \setminus \{\gamma, \eta\}}|}{|ssd_{\Gamma \setminus \{\gamma\}}||ssd_{\Gamma \setminus \{\eta\}}|}. \quad (13)$$

In the mixed case, the likelihood ratio statistic raised to the power  $2/n$  corresponding to  $\delta \perp\!\!\!\perp \gamma | Q$  is (Lauritzen, 1996, pg. 194):

$$\Lambda_{\delta\gamma.Q} = \frac{|ssd_\Gamma||ssd_{\Gamma^*}(\Delta^*)|}{|ssd_{\Gamma^*}||ssd_\Gamma(\Delta^*)|}, \quad (14)$$

where  $\Gamma^* = \Gamma \setminus \{\gamma\}$  and  $\Delta^* = \Delta \setminus \{\delta\}$ . Tur and Castelo (2011) show that for decomposable mixed GMMs, likelihood ratios in (13) and (14) follow exactly the beta distributions:

$$\Lambda_{\gamma\eta.Q} \sim \mathcal{B} \left( \frac{n - |\Gamma| - |\mathcal{I}| + 1}{2}, \frac{1}{2} \right), \text{ and}$$

$$\Lambda_{\delta\gamma.Q} \sim \mathcal{B} \left( \frac{n - |\Gamma| - |\mathcal{I}| + 1}{2}, \frac{|\mathcal{I}_{\Delta^*}|(|\mathcal{I}_\delta| - 1)}{2} \right), \quad (15)$$

respectively. This provides a proper control of the Type-I error when  $q = |Q|$  grows and allows one to define a quantity called the non-rejection rate (NRR) for mixed GMMs (Tur and Castelo, 2011) corresponding to a linear measure of association over all marginal distributions of size  $q < (n - 2)$ . The NRR is calculated for every pair of variables and is based on the outcome of a moderate number of these conditional independence tests for different  $Q$  subsets of size  $q$  sampled uniformly at random; see (Castelo and Roverato, 2006) for further details.

Using marginal distributions of size  $(q + 2) < n$  facilitates the use of complete-case analysis by replacing  $\mathcal{I}_A$  in Eq. (7) by  $\mathcal{I}_A^{obs}$  such that  $\mathcal{I}_A^{obs} \subseteq \mathcal{I}_A$  contains only the combined levels from  $A \cap \Delta$  that are fully observed. Analogously, for missing continuous values, we replace  $y^{(\nu)}$  by their observed components  $y_{obs}^{(\nu)}$  in Eqs. (2), (3) and, (4).

### 4 An EM algorithm for the Exact Conditional Independence Test

The expectation-maximization (EM) algorithm (Dempster et al., 1977) is a method to find the MLEs in statistical models when these models depend on unobserved latent variables. In the context of GMMs with  $n \gg p$ , Didelez and Pigeot (1998) use the EM-algorithm to provide MLEs of a mixed GMM

with missing values under the MAR assumption. Later, Geng et al. (2000) developed a more efficient EM algorithm (PIEM) for decomposable mixed GMMs.

Here we deal with observations that follow a homogeneous CG-distribution where discrete and/or continuous r.v. are allowed to have missing values. We denote by  $x_{obs} = (i_{obs}, y_{obs})$  the non-missing components of an observation from such distribution.

Our learning approach is based on calculating the NRR for each pair of variables  $(\alpha, \beta)$ . This requires performing conditional independence tests by computing a likelihood ratio between a saturated and a constrained decomposable model. Such a ratio only involves the *ssd* matrices derived from the decomposition of both models; see Eqs. (13), (14). Therefore, our procedure applies the EM algorithm to the saturated model to obtain the *ssd* matrix estimate corresponding to the r.v. in  $\{\alpha, \beta, Q\}$ .

Under the constrained model, however, if at each observation where  $\beta$  is non-missing, the values in the separator are also observed, Geng et al. (2000) show that the decomposition  $\{\alpha, \beta, Q\}$  is lossless. In that case, we can apply the EM algorithm separately to the clique  $C_1 = \{\alpha, Q\}$  using all the observations of the data set, denoted by  $\mathcal{X}$ , and to the clique  $C_2 = \{\beta, Q\}$  and the separator  $S = \{Q\}$  using only those observations where  $\beta$  is observed, denoted by  $\mathcal{X}^\beta$ , to get the corresponding *ssd* matrix estimates. If the constrained model cannot be decomposed losslessly, then we first compute the expectation of the sufficient statistics in the separator  $S$  using only those observations required to obtain the lossless decomposition. The detailed algorithm is as follows.

**Initialization:** First, the moment parameters of the model  $\{p_0(i), \mu_0(i), \Sigma_0\}$  are initialized. We use  $p_0(i) = |\mathcal{I}|/n$  for each  $i \in \mathcal{I}$ ,  $\mu_0(i) = 0$  for each  $\gamma \in \Gamma$  and,  $\Sigma_0 = I_{|\Gamma|}$  (identity matrix). However, in our experience, this particular algorithm is not very sensitive to other choices of the initial parameters.

**E-step:** The E-step computes the expectation of the sufficient statistics (1), (2) and, (4) given

the observed data and the current estimation of the parameters for each model. A lossless decomposition is obtained using the strategy described before and we perform the E-step to the clique  $C_2$  using the observations  $\mathcal{X}^\beta$  and to the  $C_1$  using all the observations  $\mathcal{X}$ :

$$E\{n(i_d)|x_{obs}\} = \sum_{\nu=1}^n pr(I_d = i_d|x_{obs}^{(\nu)}) = \sum_{\nu=1}^n \sum_{i' \in \mathcal{I}: i'_d = i_d} pr(I = i'|x_{obs}^{(\nu)})$$

where

$$pr(I = i'|x_{obs}^{(\nu)}) = \frac{\exp k(i')}{\sum_{s \in \mathcal{S}} \exp k(s)} \quad (16)$$

and

$$k(i) = y_{obs}^T \Sigma_{\{obs, obs\}}^{-1} \mu(i)_{obs} - \frac{1}{2} [y_{obs}^T \Sigma_{\{obs, obs\}}^{-1} y + \mu(i)_{obs}^T \Sigma_{\{obs, obs\}}^{-1} \mu_{obs}(i)] + \log p(i).$$

The set  $\mathcal{S} = \{(i_{obs}, i_{mis})|i_{mis} \in \mathcal{I}_{mis}\}$  in (16) is the set of all combinations of discrete levels given the observed ones.

$$E\{s(i_d)_\gamma|x_{obs}\} = \sum_{\nu=1}^n pr(I_d = i_d|x_{obs}^{(\nu)}) E(Y_\gamma|y_{obs}^{(\nu)}, i_d)$$

$$E\{ss(i_d)_\gamma|x_{obs}\} = \sum_{\nu=1}^n pr(I_d = i_d|x_{obs}^{(\nu)}) \times \\ [E(Y_\gamma|y_{obs}^{(\nu)}, i_d)^2 + c_{\gamma\gamma}]$$

$$E\{ss(i_d)_{\gamma,\eta}|x_{obs}\} = \sum_{\nu=1}^n pr(I_d = i_d|x_{obs}^{(\nu)}) \times \\ \{E(Y_\gamma|y_{obs}^{(\nu)}, i_d) E(Y_\eta|y_{obs}^{(\nu)}, i_d) + c_{\gamma\eta}\}$$

where

$$E(Y_\gamma|y_{obs}^{(\nu)}, i_d) = \mu(i)_\gamma - \\ \Sigma_{\{\gamma, obs\}} \Sigma_{\{obs, obs\}}^{-1} \{y_{obs} - \mu(i)_{obs}\}$$

and

$$c_{\gamma\eta} = cov(Y_{\gamma,\eta}|y_{obs}, i) = \Sigma_{\{(\gamma,\eta), (\gamma,\eta)\}} - \\ \Sigma_{\{(\gamma,\eta), obs\}} \Sigma_{\{obs, obs\}}^{-1} \Sigma_{\{obs, (\gamma,\eta)\}}$$

**M-step:** This step updates the new estimates maximizing the conditional expectations of the sufficient statistics found in the E-step. For the saturated model we compute the parameters of Eq. (9) whereas for the constrained model we only have to compute Eqs. (10), (11) and, (12).

**Convergence criteria:** For the constrained model, we first have to transform the canonical parameters to moment parameters:

$$\hat{\mu}(i) = \hat{K}^{-1}\hat{h}(i) \quad \text{and} \quad \hat{\Sigma} = \hat{K}^{-1}.$$

Then, in order to check the convergence of each model, the E-step and M-step are iterated until the maximum of

$$\left\{ \frac{|\mathcal{D}(\hat{n}(i))|}{\sqrt{(\hat{n}(i)+1)}}, \frac{|\mathcal{D}(\hat{\mu}(i)_\gamma)|}{\sqrt{\hat{\sigma}_{\gamma\gamma}}}, \frac{|\mathcal{D}(\hat{\sigma}_{\gamma\eta})|}{\sqrt{\hat{\sigma}_{\gamma\gamma}\hat{\sigma}_{\eta\eta} + (\hat{\sigma}_{\gamma\eta})^2}} \right\}$$

for  $i \in \mathcal{I}$  and  $\gamma, \eta \in \Gamma$  is smaller than a tolerance value  $\epsilon$ , for instance  $\epsilon = 0.01$ . In the numerators,  $\mathcal{D}$  denotes the difference operator between moment parameters of two consecutive iterations.

Once the convergence criterion is achieved, what we are interested in, is the last update of the *ssd* matrices we calculate at each iteration to perform the M-step.

## 5 Results

In this section we show experimental results focused on the case of mixed interactions with discrete missing values. This is, in fact, the more common situation when inferring associations between continuous gene expression profiles and discrete genotypes.

### 5.1 Synthetic Homogeneous Mixed GMM data

We build synthetic homogeneous mixed GMMs as it is done in (Tur and Castelo, 2011). First, in order to bound the size of any minimal subset separating every pair of vertices we sample an undirected  $d$ -regular graph (Harary, 1969) structure  $G = (V, E)$  with  $V = \Delta \cup \Gamma$  and excluding edges  $(i, j) \in \Delta \times \Delta$ . Then, we generate random parameters according to the conditional independences encoded in  $G$  and rendering all pairs  $(i, j) \in \Delta \times \Delta$  marginally independent.

In fact, we generate two sets of parameters: one in which the interactions between a discrete and a continuous r.v. (mixed linear interactions) are strong and another in which these interactions are weak. To generate the first model, joint levels of discrete r.v. are assigned with a uniform distribution, a nominal mean Pearson correlation  $\rho = 0.5$  is employed to generate the random covariance matrix  $\Sigma$ , and a standard deviation value  $\sigma = 3$  is set for sampling mixed linear interaction parameters  $h(i)$  with a strong effect such that  $\mu(i) = \Sigma \cdot h(i)$ . To build the model with weak mixed linear interactions, joint levels of discrete r.v. and  $\Sigma$  are generated as in the previous case but now the parameter  $h(i)$  is generated using a smaller standard deviation  $\sigma = 1.5$ . Finally, from each set of parameters we generate  $n$  observations to build the corresponding data set  $\mathcal{X} = \{x^{(\nu)}\} = \{(i^{(\nu)}, y^{(\nu)})\}$ ,  $\nu = 1, \dots, n$ , by first sampling, for every observation, a joint level  $i \in \mathcal{I}$  according to their (uniform) probability distribution, and, secondly, by sampling a multivariate normal observation from  $\mathcal{N}_{|\Gamma|}(\mu(i), \Sigma)$  for the continuous r.v.  $Y$ .

### 5.2 Synthetic Missing data

Missing values are generated under two different assumptions. We simulate, in one hand, data under the MCAR mechanism by which the occurrence of a missing value does not depend on other observed or unobserved values. On the other hand, we simulate missing data under the less stringent, and more realistic, MAR assumption. In this case, whether a value is missing or not depends on other observed values but not on unobserved ones; see (Little and Rubin, 2002).

Given a complete data set  $\mathcal{X}$  sampled as described in the previous subsection, we create a missing indicator (MI) variable  $M_\delta$  for each  $\delta \in \Delta$ . When  $M_{\delta k} = \text{TRUE}$ , then we remove the discrete value  $i^{(k)}$  from variable  $\delta$  in observation  $k$  of data set  $\mathcal{X}$ . Values for MI variables are sampled using the following logistic model (Greenland and Finkle, 1995):

$$\Pr(M_\delta = \text{TRUE}|Y = y) = \text{expit}(a + by) \quad (17)$$

where  $\text{expit}(x) = \exp(x)/(1 + \exp(x))$ . The in-

tercept is set to  $a \sim \mathcal{N}(\Phi^{-1}(\tau), 1)$ , where  $\Phi$  is the cumulative distribution function of a standard normal distribution, so that the number of missing values for each discrete r.v. equals  $\tau n$  where  $\tau$  is the desired fraction of missing values. In this paper we consider a maximum  $\tau = 0.2$  as genotypes with higher missing rates are generally discarded (The International HapMap Consortium, 2007). The coefficient  $b$  determines the dependence of  $M_\delta$  on  $Y$  such that with  $b = \ln(1)$  we obtain missing values under the MCAR assumption while  $b = \ln(3)$  produces a strong MAR effect with a 3-fold increase in the odds of missing data for each unit increase in  $Y$ .

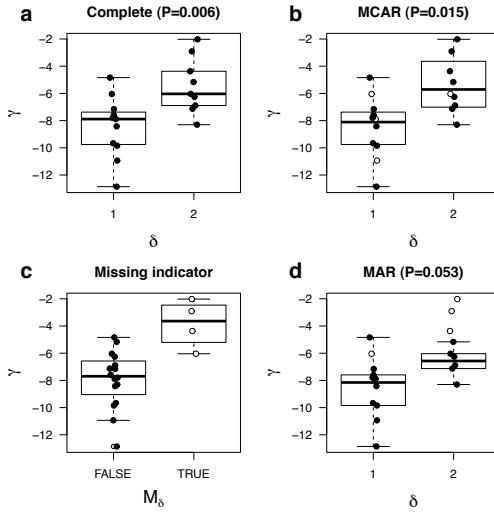


Figure 1: (a) Complete data without missing values. (b) MCAR data. (c)  $M_\delta$  depends on  $\gamma$  with  $b = \ln(2)$ . (d) MAR data using  $M_\delta$  from (c). White dots correspond to observations with missing discrete values while black dots to non-missing observations. Boxplots in (a, b, d) are calculated from non-missing observations (black dots). P indicates  $p$ -value for  $H_0 : \delta \perp\!\!\!\perp \gamma | \emptyset$ .

We have simulated  $n = 20$  observations from a correlated pair of discrete and continuous r.v.  $(\delta, \gamma)$  obtaining the association shown in Figure 1a, with a fraction  $\tau = 0.2$  of missing values under the MCAR (b) and MAR (c, d) mechanisms. Figure 1 clearly shows how the MAR effect in (c) eliminates the association between  $\delta$  and  $\gamma$  in (d) at a significance level  $\alpha = 0.05$ .

### 5.3 Analysis of Statistical Power

In this section we assess how the MCAR and MAR assumptions affect the statistical power to detect strong and weak mixed linear associations using complete-case analysis and the EM-algorithm in the exact conditional independence test.

For this purpose, we simulate 2 mixed GMMs where  $|\Delta| = |\Gamma| = 1$  and  $\delta$  and  $\gamma$  are connected so that the association between them in one model is strong and in the other, the association is weak. We sample 10,000 data sets  $\{\mathcal{X}^s, \mathcal{X}^w\}$  of  $n = 100$  observations from the strong and the weak model. From each previous data set, we simulate a MCAR data set and a MAR data set with strong effect  $b = \ln(3)$  according to missing values rates  $\tau = \{0.2, 0.15, 0.1, 0.05\}$ . We perform conditional independence tests between  $\delta$  and  $\gamma$  from the original complete data sets and from the data sets with missing values using the EM-algorithm and complete-case analysis. Finally, we calculate the statistical power as  $1 - \beta$  where  $\beta$  is the type-II error using a theoretical quantile  $\alpha = 0.05$ .

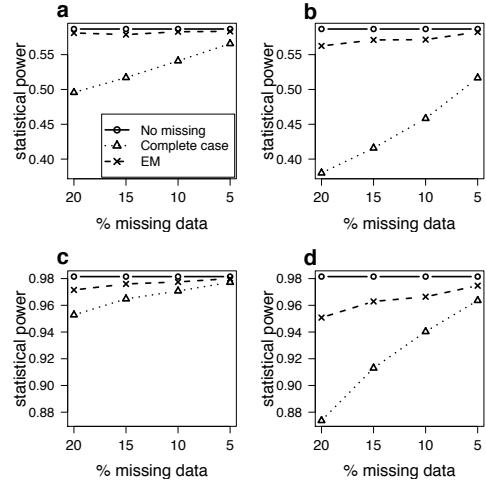


Figure 2: Statistical power to detect a mixed linear association  $(\delta, \gamma)$ . (a-b) Weak association, (c-d) strong association; (a-c) obtained under MCAR and (b-d) under a MAR effect.

In Figure 2 we assess the statistical power in each of the previously described scenarios. We

observe that EM algorithm yields higher power in front of weak and strong associations, than complete-case analysis even when missing data is sampled according to the MCAR mechanism.

#### 5.4 Accuracy of Likelihood Ratio Estimates

We want to assess the accuracy of the estimation of the likelihood ratio when it is calculated with both, the EM-algorithm and complete-case analysis.

To that end, we simulate 1,000 data sets of  $n = 30$  observations with strong mixed linear associations  $\mathcal{X}^s$  from a common graph  $G$  with  $d = 3$ ,  $|\Delta| = 3$ , and  $|\Gamma| = 47$ . For each data set, we select a pair of mixed discrete and continuous r.v.  $(\delta, \gamma)$  uniformly at random between present and absent edges. We remove 20% of values of  $\delta$  under the MCAR ( $b = \ln(1)$ ) and MAR ( $b = \ln(3)$ ) mechanisms.

For each complete and missing data sets we calculate the likelihood ratio in Eq. (14) for  $\delta \perp\!\!\!\perp \gamma | Q$  where  $Q$  is formed by variables indexed by vertices adjacent to  $\gamma$  in  $G$ .

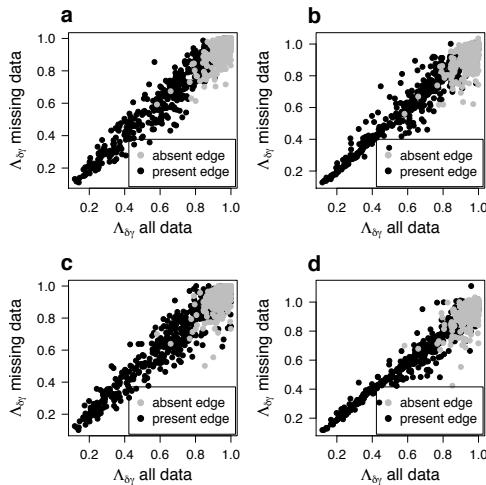


Figure 3: Comparison of likelihood ratio from a complete and a missing data set. (a-b) MCAR, (c-d) MAR; (a-c) complete-case, (b-d) EM-algorithm.

In Figure 3 we observe that estimates of likelihood ratios for present edges with complete-case analysis are significantly more variable than

those obtained with the EM-algorithm under both, MCAR and MAR assumptions (one-sided F-test of equality of variances,  $p$ -value  $< 0.05$ ).

#### 5.5 Analysis of Non-rejection Rate

To conclude this section, we want to assess the performance of using different strategies to learn a mixed GMM from data with  $p \gg n$  and missing values by means of estimating the NRR. We sample a single data set with  $n = 40$  observations from a  $d$ -regular graph with strong mixed linear interactions on  $p = 400$  and  $d = 20$ , and where  $|\Delta| = 10$ ,  $|\Gamma| = 390$ . We estimate NRR values for each pair of mixed vertices using conditioning sets of size  $q = 21$ .

In this case, for each discrete variable  $\delta \in \Delta$  we generate  $M_\delta$  according to the following model:

$$\Pr(M_\delta = \text{TRUE} | Y = (y_1, \dots, y_{|\Gamma|})) = \expit\left(a + b \sum_{i=1}^{|\Gamma|} \frac{y_i}{|\Gamma|}\right)$$

with  $a \sim \mathcal{N}(\Phi^{-1}(0.2), 1)$  and  $b = \ln(3)$ .

We estimate NRR values using complete-case analysis and the EM-algorithm with  $\epsilon = 0.001$ . Moreover, we also use the multiple imputation method of Broman et al. (2003), implemented through the `sim.gen()` function from the `R/qt1` package, that uses a hidden Markov model to impute missing genotypes in experimental crosses. We use it here to create 10 imputed data sets from the simulated data set with missing values. Then, the NRR is calculated from each of these imputed data sets and averaged at each pair of vertices.

In Figure 4, we show precision-recall curves using the NRR values from all possible mixed pairs  $(\delta, \gamma)$  with  $\delta \in \Delta$  and  $\gamma \in \Gamma$  for each method. We can observe that NRRs calculated from multiply-imputed data sets perform worse than using complete-case analysis and the EM algorithm. However, between these two latter approaches, even though EM is slightly better, there are no substantial differences in performance, as opposed to what we previously observed when we carefully assessed statistical power and accuracy.

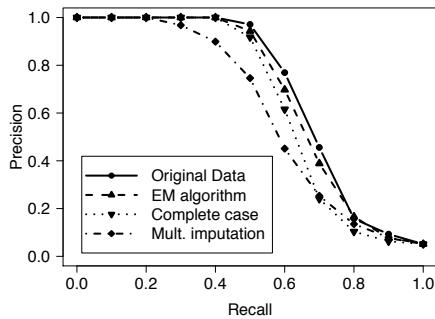


Figure 4: Precision recall curves from NRR values estimated with the original and missing data. In the latter case, the EM algorithm, complete-case analysis and multiple-imputation methods were used.

## 6 Discussion

In this paper we have introduced a statistical procedure to learn mixed GMMs from data with  $p \gg n$  and missing values, based on complete-case analysis and the EM algorithm. Our experimental results show that, for the purpose of a conditional independence test, the EM-algorithm yields more accurate estimates of the likelihood ratio for the presence of a mixed interaction (Fig. 3), and higher statistical power to detect it (Fig. 2), compared to complete-case analysis under both, the MCAR and MAR assumptions for missing data.

Intriguingly, both methods perform similarly when using the NRR to learn the structure of a mixed GMM (Fig. 4) and more research will be necessary to elucidate the circumstances under which these methods perform differently in the context of structure learning with missing data.

**Availability:** The described algorithms are implemented in the `qpCItest()` function from the Bioconductor project package `qpgraph`.

## Acknowledgments

This work is supported by a project grant [TIN2011-22826] and part of it was developed while the first author, supported by a FPI pre-doctoral fellowship [BES-2009-024901], was visiting the Dept. of Statistics at Oxford University, funded by a short-term visit fellowship

[EEBB-2011-43932], all funded by the Spanish Ministerio de Economía y Competitividad.

## References

- K. W. Broman, H. Wu, S. Sen, and G. A. Churchill. 2003. R/qtl: QTL mapping in experimental crosses. *Bioinformatics*, 19:889–890.
- R. Castelo and A. Roverato. 2006. A robust procedure for gaussian graphical model search from microarray data with  $p$  larger than  $n$ . *J Mach Learn Res*, 7:2621–2650.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J R Stat Soc B*, 39(1):1–38.
- V. Didelez and I. Pigeot. 1998. Maximum likelihood estimation in graphical models with missing values. *Biometrika*, 85(4):960–966.
- D. Edwards, G. de Abreu, and R. Labouriau. 2010. Selecting high-dimensional mixed graphical models using minimal AIC or BIC forests. *BMC Bioinformatics*, 11(1):18.
- D. Edwards. 2000. *Introduction to graphical modelling*. Springer.
- Z. Geng, K. Wan, and F. Tao. 2000. Mixed graphical models with missing data and the partial imputation EM algorithm. *Scand J Stat*, 27(3):433–444.
- S. Greenland and W. D. Finkle. 1995. A critical look at methods for handling missing covariates in epidemiologic regression analyses. *Am J Epidemiol*, 142(12):1255–1264.
- F. Harary. 1969. *Graph theory*. Addison-Wesley.
- S. Lauritzen and N. Wermuth. 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Ann Stat*, 17(1):31–57.
- S. Lauritzen. 1996. *Graphical Models*. Oxford University Press.
- R. J. A. Little and D. B. Rubin. 2002. *Statistical Analysis With Missing Data*. Probability and Statistics. Wiley, second edition.
- The International HapMap Consortium. 2007. A second generation human haplotype map of over 3.1 million SNPs. *Nature*, 449:851–861.
- I. Tur and R. Castelo. 2011. Learning mixed graphical models from data with  $p$  larger than  $n$ . In *Proc. 27th Conf. Uncertainty in Artificial Intelligence (UAI)*, pages 689–697, Barcelona, Spain.

# Non-Informative Dirichlet Score for learning Bayesian networks

Maomi Ueno

University of Electro-Communications, Japan

ueno@ai.is.uec.ac.jp

Masaki Uto

University of Electro-Communications, Japan

uto\_masaki@ai.is.uec.ac.jp

## Abstract

Learning Bayesian networks is known to be highly sensitive to the chosen equivalent sample size (ESS) in the Bayesian Dirichlet equivalence uniform (BDeu). This sensitivity often engenders unstable or undesired results because the prior of BDeu does not represent ignorance of prior knowledge, but rather a user's prior belief in the uniformity of the conditional distribution. This paper presents a proposal for a non-informative Dirichlet score by marginalizing the possible hypothetical structures as the user's prior belief. Some numerical experiments demonstrate that the proposed score improves learning accuracy. The results also suggest that the proposed score might be effective especially for small samples.

## 1 Introduction

Marginal likelihood (ML) (using a Dirichlet prior) that ensures likelihood equivalence, the most popular learning score for Bayesian networks, finds the maximum a posteriori (MAP) structure (Buntine, 1991; Heckerman *et al.*, 1995). This score is known as "Bayesian Dirichlet equivalence (BDe)" (Heckerman *et al.*, 1995). Given no prior knowledge, the Bayesian Dirichlet equivalence uniform (BDeu), as proposed earlier by Buntine (1991), is often used. Actually, BDe(u) requires an "equivalent sample size (ESS)", which reflects the degree of a user's prior belief. Moreover, recent studies have demonstrated that learning Bayesian networks is highly sensitive to the chosen equivalent sample size (ESS) (Steck and Jaakkola, 2002; Silander, Kontkanen and Myllymaki, 2007).

To clarify the BDe(u) mechanism, Ueno (2010) analyzed log-BDe(u) asymptotically, obtaining the result that it is decomposed into the log-likelihood and the penalty term of the complexity, which reflects the difference between the learned structure from data and the hypothetical structure from the user's knowledge. As the

two structures become equivalent, the penalty term is minimized with the fixed ESS. Conversely, the term increases to the degree that the two structures become different. Furthermore, the result suggests that a tradeoff exists between the role of ESS in the log-likelihood (which helps to block extra arcs) and its role in the penalty term (which helps to add extra arcs). That tradeoff might cause the BDeu score to be highly sensitive to ESS. It might make it more difficult to determine an approximate ESS.

Moreover, Ueno (2011) showed that the prior of BDeu does not represent ignorance of prior knowledge, but rather a user's prior belief in the uniformity of the conditional distribution. This fact is particularly surprising because it had been believed that BDeu has a non-informative prior. The results further imply that the optimal ESS becomes large/small when the empirical conditional distribution becomes uniform/skewed. This main factor underpins the sensitivity of BDeu to ESS.

To solve this problem, Silander, Kontkanen and Myllymaki (2007) proposed a learning method to marginalize the ESS of BDeu. They averaged BDeu values with increasing ESS

from 1 to 100 by 1.0. Moreover, to decrease the computation costs of the marginalization, Cano et al. (2011) proposed averaging a wide range of different chosen ESSs:  $ESS > 1.0$ ,  $ESS \gg 1.0$ ,  $ESS < 1.0$ ,  $ESS \ll 1.0$ . They reported that the proposed method performed robustly and efficiently.

However, such approximated marginalization does not always guarantee robust results when the optimal ESS is extremely small or large. In addition, the exact marginalization of ESS is difficult because ESS is a continuous variable in domain  $(0, \infty)$ .

Our proposal in this paper is a full non-informative Dirichlet score. We assume all possible hypothetical structures as a prior belief of BDe because the problem of BDe is that it assumes only a uniform distribution as a prior belief. This paper presents a proposal for a non-informative Dirichlet score by averaging the hypothetical structures as a user's prior belief.

The optimal ESS of BDeu becomes large/small when the empirical conditional distribution becomes uniform/skewed because its hypothetical structure assumes a uniform conditional probabilities distribution and the ESS adjusts the magnitude of the user's belief for a hypothetical structure. However, the ESS of full non-informative prior is expected to work effectively as actual pseudo-samples to augment the data, especially when the sample size is small, regardless of the uniformity of empirical distribution. This is a unique feature of the proposed method because the previous non-informative methods exclude the ESS from the score(Averaged BDeu(Silander, Kontkanen and Myllymaki, 2007,Cano et al., 2011), Normalized Maximum Likelihood (NML)(Silander, Roos, and Myllymaki, 2010)).

Some numerical experiments demonstrate that the proposed score improves learning accuracy. The results also suggest that the proposed score might be effective especially for small samples.

## 2 Learning Bayesian networks

Let  $\{x_1, x_2, \dots, x_N\}$  be a set of  $N$  discrete variables, each of which can take a value in the set of

states  $\{1, \dots, r_i\}$ . Here,  $x_i = k$  means that an  $x_i$  is state  $k$ . According to the Bayesian network structure  $g \in G$ , the joint probability distribution is given as

$$p(x_1, x_2, \dots, x_N | g) = \prod_{i=1}^N p(x_i | \Pi_i, g), \quad (1)$$

where  $G$  signifies the possible set of Bayesian network structures, and where  $\Pi_i$  denotes the parent variables set of  $x_i$ .

Next, we introduce the problem of learning a Bayesian network. Let  $\theta_{ijk}$  be a conditional probability parameter of  $x_i = k$  when the  $j$ -th instance of the parents of  $x_i$  is observed (We write  $\Pi_i = j$ ). Buntine (1991) assumed the Dirichlet prior and used an expected a posteriori (EAP) estimator as the parameter estimator  $\hat{\Theta} = (\hat{\theta}_{ijk}), (i = 1, \dots, N, j = 1, \dots, q_i, k = 1, \dots, r_i - 1)$ :

$$\hat{\theta}_{ijk} = \frac{\alpha_{ijk} + n_{ijk}}{\alpha_{ij} + n_{ij}}, (k = 1, \dots, r_i - 1), \quad (2)$$

where  $n_{ijk}$  represents the number of samples of  $x_i = k$  when  $\Pi_i = j$  and  $n_{ij} = \sum_{k=1}^{r_i} n_{ijk}$ , and where  $\alpha_{ijk}$  denotes the hyperparameters of the Dirichlet prior distributions. ( $\alpha_{ijk}$  is a pseudo-sample corresponding to  $n_{ijk}$ ),  $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$ , and  $\hat{\theta}_{ijr_i} = 1 - \sum_{k=1}^{r_i-1} \hat{\theta}_{ijk}$ .

The marginal likelihood is obtained as

$$p(\mathbf{X} | \alpha, g) = \prod_{i=1}^N \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})}. \quad (3)$$

Here,  $q_i$  signifies the number of instances of  $\Pi_i$  in which  $q_i = \prod_{x_l \in \Pi_i} r_l$ . In addition,  $\mathbf{X}$  is a dataset. The problem of learning a Bayesian network is to find the MAP structure that maximizes the score (3).

Particularly, Heckerman et al. (1995) presented a sufficient condition for satisfying the likelihood equivalence assumption, which says that data should not help to discriminate network structures that represent the same assertions of conditional independence, in the form of the following constraint related to hyperparameters of (3):

$$\sum_{i=1}^N \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \alpha_{ijk} = const. \quad (4)$$

Furthermore, they proposed a marginal likelihood that reflects a user's prior knowledge as

shown below.

$$p(\mathbf{X} | \alpha, g, g^h) = \prod_{i=1}^N \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ijk}^{g^h})}{\Gamma(\alpha_{ij}^{g^h} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}^{g^h} + n_{ijk})}{\Gamma(\alpha_{ijk}^{g^h})} \quad (5)$$

$$\alpha_{ijk}^{g^h} = \alpha p(x_i = k, \Pi_i^g = j | g^h) \quad (6)$$

Here,  $\alpha$  is the user-determined equivalent sample size (ESS),  $\Pi_i^g$  denotes the parent variable sets of  $x_i$  according to  $g$  and  $g^h$  is the hypothetical Bayesian network structure that reflects a user's prior knowledge. This metric was designated as the Bayesian Dirichlet equivalence (BDe) score metric.

As Buntine (1991) described,  $\alpha_{ijk}^{g^h} = \frac{\alpha}{(r_i q_i)}$  is regarded as a special case of the BDe metric. Heckerman *et al.* (1995) designated this special case as "BDe".

For cases of which we have no prior knowledge, BDeu is often used in practice. Heckerman *et al.* (1995) reported, as a result of their comparative analyses of BDeu and BDe, that BDeu is better than BDe unless the user's beliefs are close to the true model. BDeu requires an "equivalent sample size (ESS)", which is the value of a free parameter specified by the user. Recent reports have described that ESS in BDeu plays an important role in learning Bayesian networks (Steck and Jaakkola, 2002; Silander, Kontkanen and Myllymaki, 2007).

Especially, Ueno (2011) reported that the prior of BDeu does not represent ignorance of prior knowledge but rather a user's prior belief in the uniformity of the conditional distribution. This result is particularly surprising because it had been believed that BDeu has a non-informative prior. In addition, he explained the mechanism by which the optimal ESS becomes large/small when the empirical conditional distribution becomes uniform/skewed because ESS determines the magnitude of the user's prior belief for a hypothetical structure. This mechanism causes the BDeu score to be highly sensitive to ESS.

### 3 Non-informative Dirichlet Score

This section presents an alternative non-informative prior Dirichlet score because BDeu has no non-informative prior.

To clarify the mechanism of BDe, Ueno (2010) analyzed the log-BDe asymptotically and derived the following theorem.

**Theorem 1.** (Ueno 2010) When  $\alpha + nt$  is sufficiently large, log-BDe converges to

$$\log p(\mathbf{X} | \alpha, g, g^h) = \log p(\widehat{\Theta}^g | \mathbf{X}, \alpha, g, g^h) \quad (7)$$

$$-\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{r_i - 1}{r_i} \log \left( 1 + \frac{n_{ijk}}{\alpha_{ijk}^{g^h}} \right) + \text{const},$$

$$\text{where } \log p(\widehat{\Theta}^g | \mathbf{X}, \alpha, g, g^h) =$$

$$\sum_{i=1}^N \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} (\alpha_{ijk}^{g^h} + n_{ijk}) \log \frac{(\alpha_{ijk}^{g^h} + n_{ijk})}{(\alpha_{ij}^{g^h} + n_{ij})},$$

*const* is independent terms of the number of parameters, and  $\widehat{\Theta}^g = \{\widehat{\theta}_{ijk}^g\}$ , ( $i = 1, \dots, N, j = 1, \dots, q_i, k = 1, \dots, r_i - 1$ ),

$$\widehat{\theta}_{ijk}^g = \frac{\alpha_{ijk}^{g^h} + n_{ijk}}{\alpha_{ij}^{g^h} + n_{ij}}. \quad (8)$$

From (7), log-BDe can be decomposed into two factors: 1. a log-posterior term  $\log p(\widehat{\Theta}^g | \mathbf{X}, \alpha, g, g^h)$  and 2. a penalty term  $\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{r_i - 1}{r_i} \log \left( 1 + \frac{n_{ijk}}{\alpha_{ijk}^{g^h}} \right)$ .  $\sum_{i=1}^N \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{r_i - 1}{r_i}$  is the number of parameters.

This well known model selection formula is generally interpreted 1. as reflecting the fit to the data and 2. as signifying the penalty that blocks extra arcs from being added.

Ueno (2010) described that the term  $\sum_{i=1}^N \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \log \left( 1 + \frac{n_{ijk}}{\alpha_{ijk}^{g^h}} \right)$  in (7) reflects the difference between the learned structure from data and the hypothetical structure  $g^h$  from the user's knowledge in BDe. To the degree that the two structures are equivalent, the penalty term is minimized with the fixed ESS. Conversely, to the degree that the two structures differ, the term is larger. Moreover, from (7),  $\alpha$  determines the magnitude of the user's prior belief for a hypothetical structure  $g^h$ .

On the other hand, BDeu, of which the prior distribution assumes an uniform distribution of conditional probabilities, had been believed to employ a non-informative prior. However, from (7), the optimal ESS of BDeu becomes

large/small when the empirical conditional distribution becomes uniform/skewed because its hypothetical structure  $g^h$  assumes a uniform conditional probabilities distribution and the ESS adjusts the magnitude of the user's belief for a hypothetical structure. Namely, the prior of BDeu does not represent ignorance of prior knowledge but rather a user's prior belief in the uniformity of the conditional distribution.

The main purpose of this paper is to develop a Dirichlet score that has a full non-informative prior. For this purpose, we assume all possible hypothetical structures as a prior belief of BDe because a problem of BDeu is that it assumes only a uniform distribution as a prior belief.

That is, we marginalize ML over the hypothetical structures  $g^h \in G$  as follows:

**Definition 1.** *NIP – BDe* (Non-informative prior Bayesian Dirichlet equivalence) is defined as

$$p(\mathbf{X} | g, \alpha) = \sum_{g^h \in G} p(g^h) p(\mathbf{X} | \alpha, g, g^h) \quad (9)$$

$$= \sum_{g^h \in G} p(g^h) \left( \prod_{i=1}^N \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij}^{g^h})}{\Gamma(\alpha_{ij}^{g^h} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}^{g^h} + n_{ijk})}{\Gamma(\alpha_{ijk}^{g^h})} \right),$$

where  $\sum_{g^h \in G}$  is the summation over the possible hypothetical structures, and where  $p(g^h)$  is a uniform distribution.

To estimate the marginal ML, it is necessary to calculate  $p(x_i = k, \Pi_i^g = j | g^h)$  automatically for all the structures. For this purpose, we use an empirical estimation of  $\hat{p}(x_i = k, \Pi_i^g = j | g^h, \Theta^{g^h})$  using data. Here, we estimate the joint probability estimate  $\hat{p}(x_i = k, \Pi_i^g = j | g^h, \Theta^{g^h})$  which is transformed to the joint probability of  $x_i$  and its parents variables in  $g$  from the estimated conditional probability parameters  $\Theta^{g^h}$  given  $g^h$ .

Our method has the following two steps:

1. Estimate the conditional probability parameters set  $\Theta^{g^h} = \{\theta_{ijk}^{g^h}\}, (i = 1, \dots, N, j = 1, \dots, q_i, k = 1, \dots, r_i - 1)$  given  $g^h$  from data
2. Estimate the joint probability  $\hat{p}(x_i = k, \Pi_i^g = j | g^h, \Theta^{g^h})$

First, we estimate the conditional probability parameters set given  $g^h$  as

$$\widehat{\theta}_{ijk}^{g^h} = \frac{\frac{1}{r_i q_i^{g^h}} + n_{ijk}^{g^h}}{\frac{1}{q_i^{g^h}} + n_{ij}^{g^h}}, \quad (10)$$

where  $n_{ijk}^{g^h}$  represents the number of samples of  $x_i = k$  when  $\Pi_i^{g^h} = j$  (parent variables set of  $x_i$  given  $g^h$ ) and  $n_{ij}^{g^h} = \sum_{k=1}^{r_i} n_{ijk}^{g^h}$ , and  $q_i^{g^h}$  denotes the number of parent variables of  $\Pi_i^{g^h}$ .

Next, we calculate the estimated joint probability  $\hat{p}(x_i = k, \Pi_i^g = j | g^h, \Theta^{g^h})$  as shown below.

$$\begin{aligned} \hat{p}(x_i = k, \Pi_i^g = j | g^h, \Theta^{g^h}) &= \\ &\sum_{x_l \notin x_i \cup \Pi_i^g} p(x_1, \dots, x_i, \dots, x_N | g^h, \Theta^{g^h}) \end{aligned} \quad (11)$$

For the computation of (11), we can employ various marginalization algorithms (e.g. variable elimination, factor elimination, random sampling elimination: see, for example, (Darwiche, 2009)).

In practice, however, the Expected log-BDe is difficult to calculate because the product of multiple probabilities suffers serious computational problems. To avoid this, we propose an alternative method, Expected log-BDe, as described below.

**Definition 2.** Expected log-BDe is defined as

$$\begin{aligned} E_{g^h \in G} \log p(\mathbf{X} | \alpha, g, g^h) &= \\ &\sum_{g^h \in G} p(g^h) \log p(\mathbf{X} | \alpha, g, g^h) \\ &= \sum_{g^h \in G} p(g^h) \left( \sum_{i=1}^N \sum_{j=1}^{q_i} \log \frac{\Gamma(\alpha_{ij}^{g^h})}{\Gamma(\alpha_{ij}^{g^h} + n_{ij})} \right. \\ &\quad \left. \sum_{k=1}^{r_i} \log \frac{\Gamma(\alpha_{ijk}^{g^h} + n_{ijk})}{\Gamma(\alpha_{ijk}^{g^h})} \right), \end{aligned} \quad (12)$$

Compared to NIP-BDe, the Expected log-BDe is practical for computation because it can be calculated by the sum of log-BDe.

Although a similar Bayesian model averaging criterion has already been proposed (Chickering and Heckerman, 2000), (Tian, He, and Ram, 2010), its purpose is not to predict the true structure but to seek the optimal structure

which maximizes the inference prediction of a new data  $x_{i+1}$ . Therefore, their purpose is not the same as that of this study.

#### 4 Learning algorithm using dynamic programming

Learning Bayesian networks is known to be an NP complete problem (Chickering, 1996). Recently, however, the exact solution methods can produce results in reasonable computation time if the variables are not prohibitively numerous (e.g.(Silander and Myllymaki, 2006), Malone et al., 2011).

We employ the learning algorithm of (Silander and Myllymaki, 2006) using dynamic programming. Our algorithm comprises four steps:

1. Compute the local Expected log-BDe scores for all possible  $N2^{N-1}$   $(x_i, \Pi_i^g)$  pairs.
2. For each variable  $x_i \in \mathbf{x}$ , find the best parent set in parent candidate set  $\{\Pi_i^g\}$  for all  $\Pi_i^g \subseteq \mathbf{x} \setminus \{x_i\}$ ,  $(\forall g \in G)$ .
3. Find the best sink for all  $2^N$  variables and the best ordering.
4. Find the optimal Bayesian network

Only Step 1 is different from the procedure described by Silander *et al.*, 2006. First, to obtain the local Expected log-BDe score for a variable and its parent variables set pair, we should average log-BDe for all possible hypothetical structures. To compute the local Expected log-BDe score for each pair  $(x_i, \Pi_i^g)$  in Step 1, conditional frequency tables  $cft(x_i, \Pi_i^{g^h})$  for all possible hypothesis parent sets  $\{\Pi_i^{g^h}\}$  for  $(\forall g^h \in G)$  are needed.

Algorithm 1 gives a pseudo-code for computing the local Expected log-BDe scores,  $LS[x_i][\Pi_i^g]$ , for all possible  $(x_i, \Pi_i^g)$  pairs. The pseudo-code assumes some helper functions:  $getCft(x_i, \Pi_i^g)$  produces the conditional frequency table  $cft(x_i, \Pi_i^g)$ , and  $getTl(\Pi_i^g, cft[x_i][\Pi_i^{g^h}])$  translates the joint probability estimate  $\hat{p}(x_i = k, \Pi_i^g = j | g^h, \Theta^{g^h})$  from  $cft[x_i][\Pi_i^{g^h}]$ , and the function

$score_i(\Pi_i^g, Tl[x_i][\Pi_i^g][\Pi_i^{g^h}])$  calculates the local log-BDe score of  $g$  given a hypothetical structure  $g^h$ .

First,  $getCft(x_i, \Pi_i^g)$  produces the conditional frequency tables  $cft(x_i, \Pi_i^g)$  for all possible  $\Pi_i^g$ . They are stored on the memory or the disk as soon as they are produced. This procedure is the same as that of (Silander and Myllymaki, 2006). Next,  $getTl(\Pi_i^g, cft[x_i][\Pi_i^{g^h}])$  calculates the joint probability estimate  $\hat{p}(x_i = k, \Pi_i^g = j | g^h, \Theta^{g^h})$  using the stored conditional frequency tables by marginalizing the product of factors in  $\Theta^{g^h}$  including  $\{x_i, \Pi_i^g\}$ . This procedure runs in  $O(|\{x_i \cup \Pi_i^g\}| \exp(w))$  given an elimination order of width  $w$ . The local Expected log-BDe,  $LS[x_i][\Pi_i^g]$ , can be calculated as the summation of local log-BDe values for all local possible structures  $\Pi_i^{g^h}$ .

---

#### Algorithm 1 $getLocalScore(\mathbf{x})$ .

---

```

for all $x_i \in \mathbf{x}$ do
 for all $\{\Pi_i^g\} \subseteq \mathbf{x} \setminus \{x_i\}$ do
 if $cft[x_i][\Pi_i^g] = null$ then
 $cft[x_i][\Pi_i^g] \leftarrow get Cft(x_i, \Pi_i^g)$
 end if
 end for
 for all $\{\Pi_i^g\} \subseteq \mathbf{x} \setminus \{x_i\}$ do
 for all $\{\Pi_i^{g^h}\} \subseteq \mathbf{x} \setminus \{x_i\}$ do
 $Tl[x_i][\Pi_i^g][\Pi_i^{g^h}] \leftarrow get Tl(\Pi_i^g, cft[x_i][\Pi_i^{g^h}])$
 $LS[x_i][\Pi_i^g] \leftarrow LS[x_i][\Pi_i^g] + score_i(\Pi_i^g, Tl[x_i][\Pi_i^g][\Pi_i^{g^h}])$
 end for
 $LS[x_i][\Pi_i^g] \leftarrow LS[x_i][\Pi_i^g] / |\{\Pi_i^g \subseteq \mathbf{x} \setminus \{x_i\}\}|$
 end for
end for
if $|\mathbf{x}| > 1$ then
 $getLocalScore(\mathbf{x} \setminus \{x_i\})$
end if

```

---

The time complexity of the Algorithm 1 is  $O(N^2 2^{2(N-1)} \exp(w))$ . After computing the local scores, we find the optimal Bayesian network in Steps 2, 3 and 4 of our algorithm. Steps 2–4 are the same as those proposed by Silander *et al.* (2006). Although the traditional scores (BDeu, AIC, BIC, and so on) run in  $O(N2^{(N-1)})$ , the proposed method requires greater computation costs. However, the required memory for the proposed computation is equal to that of the traditional scores.

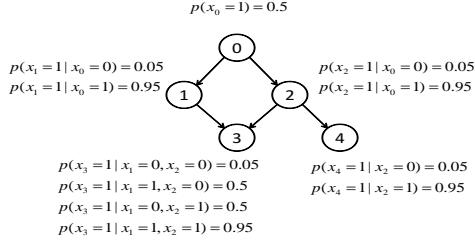


Figure 1: g1: Strongly skewed distribution.

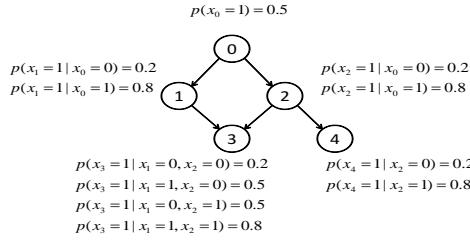


Figure 2: g2: Skewed distribution.

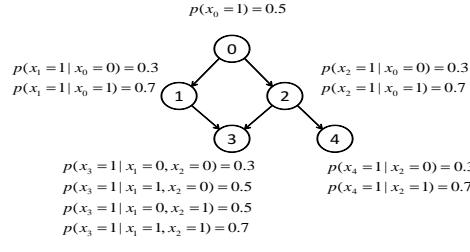


Figure 3: g3: Uniform distribution.

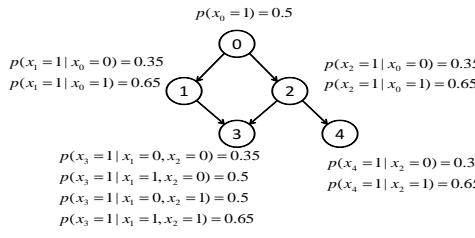


Figure 4: g4: Strongly uniform distribution.

## 5 Simulation experiments

We conducted simulation experiments to compare Expected log-BDe with BDeu according to the procedure described by Ueno (2011). We used small network structures with binary variables in Figs. 1, 2, 3, and 4, in which the distributions are changed from skewed to uniform. Figure 1 presents a structure in which the conditional probabilities differ greatly because of the parent variable states (g1: Strongly skewed

Table 1: Learning performance of BDeu

| g1   |    | BDeu( $\alpha = 0.1$ ) |     |     |    |     |     |
|------|----|------------------------|-----|-----|----|-----|-----|
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 5  | 233                    | 97  | 21  | 12 | 81  | 22  |
| 500  | 76 | 67                     | 21  | 11  | 5  | 19  | 11  |
| 1000 | 98 | 5                      | 0   | 2   | 0  | 2   | 1   |
| g2   |    | BDeu( $\alpha = 0.1$ ) |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 5  | 211                    | 97  | 9   | 2  | 91  | 12  |
| 500  | 82 | 52                     | 8   | 0   | 9  | 19  | 16  |
| 1000 | 99 | 4                      | 0   | 0   | 1  | 1   | 2   |
| g3   |    | BDeu( $\alpha = 0.1$ ) |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 0  | 205                    | 131 | 3   | 0  | 69  | 2   |
| 500  | 9  | 194                    | 85  | 0   | 2  | 95  | 12  |
| 1000 | 72 | 74                     | 16  | 1   | 13 | 27  | 17  |
| g4   |    | BDeu( $\alpha = 0.1$ ) |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 0  | 253                    | 208 | 5   | 0  | 38  | 2   |
| 500  | 0  | 208                    | 111 | 1   | 3  | 86  | 7   |
| 1000 | 16 | 184                    | 77  | 1   | 3  | 88  | 15  |
| g1   |    | BDeu( $\alpha = 1.0$ ) |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 34 | 173                    | 53  | 17  | 19 | 56  | 28  |
| 500  | 84 | 52                     | 5   | 11  | 6  | 16  | 14  |
| 1000 | 96 | 10                     | 0   | 4   | 0  | 4   | 2   |
| g2   |    | BDeu( $\alpha = 1.0$ ) |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 32 | 177                    | 54  | 10  | 11 | 69  | 33  |
| 500  | 86 | 45                     | 0   | 2   | 11 | 13  | 19  |
| 1000 | 99 | 4                      | 0   | 0   | 1  | 1   | 2   |
| g3   |    | BDeu( $\alpha = 1.0$ ) |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 3  | 227                    | 101 | 8   | 5  | 86  | 27  |
| 500  | 33 | 189                    | 32  | 1   | 18 | 79  | 59  |
| 1000 | 85 | 49                     | 1   | 0   | 13 | 16  | 19  |
| g4   |    | BDeu( $\alpha = 1.0$ ) |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 1  | 233                    | 158 | 7   | 2  | 56  | 10  |
| 500  | 7  | 210                    | 84  | 1   | 9  | 93  | 23  |
| 1000 | 41 | 175                    | 28  | 0   | 15 | 75  | 57  |
| g1   |    | BDeu( $\alpha = 10$ )  |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 1  | 458                    | 17  | 223 | 11 | 123 | 84  |
| 500  | 24 | 252                    | 2   | 112 | 2  | 100 | 36  |
| 1000 | 47 | 176                    | 0   | 73  | 2  | 72  | 29  |
| g2   |    | BDeu( $\alpha = 10$ )  |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 40 | 180                    | 18  | 43  | 16 | 60  | 43  |
| 500  | 74 | 73                     | 0   | 17  | 10 | 26  | 20  |
| 1000 | 87 | 32                     | 0   | 14  | 1  | 8   | 9   |
| g3   |    | BDeu( $\alpha = 10$ )  |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 6  | 299                    | 46  | 23  | 36 | 87  | 107 |
| 500  | 44 | 200                    | 4   | 14  | 32 | 66  | 84  |
| 1000 | 84 | 50                     | 0   | 3   | 13 | 15  | 19  |
| g4   |    | BDeu( $\alpha = 10$ )  |     |     |    |     |     |
| n    | o  | SHD                    | ME  | EE  | WO | MO  | EO  |
| 200  | 5  | 290                    | 94  | 26  | 19 | 77  | 74  |
| 500  | 22 | 249                    | 36  | 10  | 30 | 82  | 91  |
| 1000 | 51 | 174                    | 4   | 3   | 24 | 60  | 83  |

distribution). By gradually reducing the difference of the conditional probabilities from Fig. 1, we generated Fig. 2 (g2: Skewed distribution), Fig. 3 (g3: Uniform distribution), and Fig. 4 (g4: Strongly uniform distribution).

Procedures used for the simulation experiments are described below.

1. We generated 200, 500, and 1,000 samples

Table 2: Learning performance of Expected log-BDe

|      |     | Expected log-BDe( $\alpha = 0.1$ ) |     |     |    |    |     |    |
|------|-----|------------------------------------|-----|-----|----|----|-----|----|
| g1   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 5   | 264                                | 93  | 40  | 19 | 78 | 34  |    |
| 500  | 62  | 98                                 | 19  | 31  | 4  | 32 | 12  |    |
| 1000 | 91  | 19                                 | 0   | 9   | 0  | 8  | 2   |    |
| g2   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 10  | 203                                | 91  | 9   | 3  | 86 | 14  |    |
| 500  | 86  | 41                                 | 4   | 1   | 8  | 14 | 14  |    |
| 1000 | 100 | 0                                  | 0   | 0   | 0  | 0  | 0   |    |
| g3   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 0   | 209                                | 122 | 4   | 0  | 78 | 5   |    |
| 500  | 20  | 175                                | 70  | 0   | 6  | 81 | 18  |    |
| 1000 | 81  | 55                                 | 8   | 1   | 12 | 18 | 16  |    |
| g4   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 0   | 247                                | 192 | 6   | 0  | 45 | 4   |    |
| 500  | 2   | 204                                | 102 | 1   | 3  | 91 | 7   |    |
| 1000 | 23  | 179                                | 66  | 1   | 6  | 82 | 24  |    |
|      |     | Expected log-BDe( $\alpha = 1.0$ ) |     |     |    |    |     |    |
| g1   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 34  | 191                                | 52  | 30  | 22 | 49 | 38  |    |
| 500  | 84  | 55                                 | 5   | 18  | 3  | 22 | 7   |    |
| 1000 | 92  | 17                                 | 0   | 8   | 0  | 8  | 1   |    |
| g2   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 45  | 147                                | 36  | 12  | 11 | 56 | 32  |    |
| 500  | 88  | 37                                 | 0   | 2   | 9  | 11 | 15  |    |
| 1000 | 100 | 0                                  | 0   | 0   | 0  | 0  | 0   |    |
| g3   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 4   | 241                                | 91  | 12  | 10 | 85 | 43  |    |
| 500  | 50  | 144                                | 20  | 1   | 20 | 55 | 48  |    |
| 1000 | 87  | 45                                 | 0   | 1   | 12 | 14 | 18  |    |
| g4   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 2   | 244                                | 141 | 10  | 4  | 62 | 27  |    |
| 500  | 11  | 223                                | 70  | 3   | 18 | 89 | 43  |    |
| 1000 | 45  | 180                                | 19  | 0   | 20 | 69 | 72  |    |
|      |     | Expected log-BDe( $\alpha = 10$ )  |     |     |    |    |     |    |
| g1   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 6   | 447                                | 24  | 215 | 13 | 87 | 108 |    |
| 500  | 68  | 112                                | 6   | 49  | 2  | 34 | 21  |    |
| 1000 | 96  | 8                                  | 0   | 4   | 0  | 4  | 0   |    |
| g2   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 51  | 126                                | 14  | 42  | 11 | 34 | 25  |    |
| 500  | 82  | 44                                 | 0   | 10  | 8  | 12 | 14  |    |
| 1000 | 91  | 22                                 | 0   | 9   | 0  | 8  | 5   |    |
| g3   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 14  | 285                                | 30  | 42  | 34 | 77 | 102 |    |
| 500  | 58  | 132                                | 1   | 20  | 21 | 38 | 52  |    |
| 1000 | 84  | 49                                 | 0   | 6   | 12 | 14 | 17  |    |
| g4   |     | o                                  | SHD | ME  | EE | WO | MO  | EO |
| 200  | 4   | 324                                | 75  | 47  | 25 | 74 | 103 |    |
| 500  | 25  | 250                                | 27  | 19  | 33 | 73 | 98  |    |
| 1000 | 55  | 158                                | 2   | 7   | 22 | 52 | 75  |    |

from the four figures.

- Using BDeu and Expected log-BDe by changing  $\alpha$  (0.1, 1.0, 10), Bayesian network structures were estimated, respectively, based on 200, 500, and 1,000 samples. We searched for the exactly true structure.

3. The times the estimated structure was the true structure (when Structural Hamming Distance (SHD) is zero; Tsamardinos, Brown, and Aliferis, 2006) were counted by repeating procedure 2 for 100 iterations.

We employ the variable elimination algorithm (Darwiche, 2009) for the marginalization in (11) because our experiments use only a five-node network. Its computational cost is not burdensome.

Table 1 presents the results for BDeu. Table 2 presents results for Expected log-BDe. Column “o” shows the number of correct-structure estimates in 100 trials. Column “SHD” shows the Structure Hamming Distance (SHD). Column “ME” shows the total number of missing arcs, column “EE” shows the total number of extra arcs, column “WO” shows the total number of arcs with wrong orientation, column “MO” shows the total number of arcs with missing orientation, and column “EO” shows the total number of arcs with extra orientation in the case of likelihood equivalence. The results presented in Table 1 reveal that BDeu is highly sensitive to  $\alpha$ .

In Table 1, the optimum value of  $\alpha$  becomes small as the conditional distribution becomes skewed. In contrast, the optimum value of  $\alpha$  becomes large as the conditional distribution becomes uniform.

As shown in Table 2, the performances of Expected log-BDe are better than those of BDeu in almost all cases. The results also show that the BDeu performances are highly sensitive to  $\alpha$  but those of Expected log-BDe are robust for  $\alpha$ .

Additionally, it is noteworthy that the performance of BDeu is extremely worse than those of Expected log-BDe for g4 with  $\alpha = 0.1$  and g1 with  $\alpha = 10$ . The reason is that the optimal  $\alpha$  becomes large/small for uniform/skewed conditional probabilities because BDeu assumes a uniform conditional prior. Although the optimal  $\alpha$  for BDeu is highly sensitive to the uniformity of conditional probabilities distribution, the optimal  $\alpha$  of the proposed method is robust for the uniformity.

Especially for small samples, the large  $\alpha$  set-

ting for the proposed method works effectively because the learning performances of  $\alpha = 10$  for  $n = 200$  are better than those of  $\alpha = 0.1$ , which means that the ESS of Expected log-BDe might be affected only by the sample size. The results also suggest that the optimal ESS increases as the sample size becomes small. Therefore, the ESS of the proposed method works effectively as actual pseudo-samples to augment the data, especially when the sample size is small.

## 6 Conclusions

This paper presented a proposal for a non-informative Dirichlet score. The results suggest that the proposed method is effective especially for a small sample size. The future tasks are the following: 1. Analysis of the proposed method using various simulation data. 2. Improvement of the learning algorithm. 3. Determination of the optimal  $\alpha$  for a given data size. Furthermore, NML (Silander, Roos, and Myllymaki, 2010) is known as an alternative information-theoretic approach for circumventing the problem with ESS. It is also an important future task to compare the performances of these non-informative learning scores.

## References

- W.L. Buntine. 1991. Theory refinement on Bayesian networks. In B. D'Ambrosio, P. Smets and P. Bonissone, (eds.), *Proc. of the Seventh Int. Conf. Uncertainty in Artificial Intelligence*, pages 52–60.
- E. Castillo, A.S. Hadi and C. Solares. 1997. Learning and updating of uncertainty in Dirichlet models. *Machine Learning*, **26**, pages 43–63.
- D. Chickering. 1995. Learning Bayesian networks is NP-complete. In *Learning from Data – Artificial Intelligence and Statistics*, V, pages 121–130.
- D.M. Chickering and D. Heckerman. 2000. A comparison of scientific and engineering criteria for Bayesian model selection. *Statistics and Computing*, **10**, pages 55–62.
- A. Cano, M. Gomez-Olmedo, A.R. Masegosa, and S. Moral. 2011. Locally Averaged Bayesian Dirichlet Metrics. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty Symbolic and Quantitative Approaches to Reasoning with Uncertainty(Ecsqaru 2011)*, pages 217–228.
- A. Darwiche. 2009. Modeling and reasoning with Bayesian networks. Cambridge University Press.
- D. Heckerman, D. Geiger and D. Chickering. 1995. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, **20**, pages 197–243.
- B.M. Malone, C. Yuan, E.A. Hansen, and S. Bridges. 2011. Improving the Scalability of Optimal Bayesian Network Learning with External-Memory Frontier Breadth-First Branch and Bound Search. In A. Prefter and F.G. Cozman (eds.) *Proc. the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 479–488.
- T. Silander and P. Myllymaki. 2006. A simple approach for finding the globally optimal Bayesian network structure. In R. Dechter and T. Richardson (eds.), *Proc. 22nd Conf. on Uncertainty in Artificial Intelligence*, pages 445–452.
- T. Silander, P. Kontkanen and P. Myllymaki. 2007. On sensitivity of the MAP Bayesian network structure to the equipment sample size parameter, In K.B. Laskey, S.M. Mahoney and J. Goldsmith (eds.), *Proc. 23rd Conference on Uncertainty in Artificial Intelligence*, pages 360–367.
- T. Silander, T. Roos and P. Myllymaki. 2010. Learning Locally Minimax Optimal Bayesian Networks. *International Journal of Approximate Reasoning*, **51**(5): 544–557.
- H. Steck and T.S. Jaakkola. 2002. On the Dirichlet Prior and Bayesian Regularization. In S. Becker, S. Thrun, and K. Obermayer (eds.), *Advances in Neural Information Processing Systems (NIPS)*, pages 697–704.
- H. Steck. 2008. Learning the Bayesian network structure: Dirichlet Prior versus Data. D.A. McAllester and P. Myllymaki (eds.), *Proc. 24th Conference on Uncertainty in Artificial Intelligence*, pages 511–518.
- J. Tian, R. He and L. Ram. 2010. Bayesian Model Averaging Using the k-best Bayesian Network Structures. In P. Grunwald and P. Spirtes (eds.), *Proc. 26th Conference on Uncertainty in Artificial Intelligence*, pages 589–597.
- I. Tsamardinos, L.E. Brown, and C.F. Aliferis. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, **65**(1), pages 1–78.
- M. Ueno. 2010. Learning networks determined by the ratio of prior and data. In P. Grunwald and P. Spirtes (eds.), *Proc. 26th Conference on Uncertainty in Artificial Intelligence*, pages 598–605.
- M. Ueno. 2011. Robust learning Bayesian networks for prior belief. In A. Prefter and F.G. Cozman (eds.) *Proc. 27th Conference on Uncertainty in Artificial Intelligence*, pages 698–707.

# New Local Move Operators for Bayesian Network Structure Learning

Jimmy Vandel and Brigitte Mangin and Simon de Givry  
UBIA, UR 875, INRA, F-31320 Castanet Tolosan, France  
[{jvandel,mangin,degivry}@toulouse.inra.fr](mailto:{jvandel,mangin,degivry}@toulouse.inra.fr)

## Abstract

We propose new local move operators incorporated into a score-based stochastic greedy search algorithm to efficiently escape from local optima in the search space of directed acyclic graphs. We extend the classical set of arc addition, deletion, and reversal operators with a new operator replacing or *swapping* one parent to another for a given node, *i.e.* combining two elementary operations (arc addition and deletion) in one move. The operators are further extended by doing more operations in one move in order to overcome the acyclicity constraint of Bayesian networks. These extra operations are temporally performed in the space of directed *cyclic* graphs, acyclicity being restored at the end and the move kept if it increases the score. Our experimental results on standard Bayesian networks and challenging gene regulatory nets show large BDeu improvements compared to state-of-the-art structure learning algorithms when the sample size is small.

## 1 Introduction

Learning the structure of Bayesian networks from fully observed data is known to be an NP-hard problem (Chickering and Heckerman, 1996) which has received a lot of attention from researchers during the last two decades (Daly et al., 2011). Due to its difficulty, heuristic methods have been widely used to learn Bayesian network structures. Among them, score-based methods use a scoring function  $f$  to score a network structure with respect to the data. They explore the space of network structures to find the highest-scoring network. This space being superexponential in the number of variables, local search methods are used such as greedy ascent search (also called hill-climbing), tabu search, simulated-annealing, and other complex metaheuristics. In spite of its simplicity, the (repeated randomized or *stochastic*) greedy search method reveals to be a competitive method compared to more complex algorithms (Gámez et al., 2011). Starting from an initial network structure, it performs a series of local moves until a local optimum is found. Each move selects and applies the best elementary operation(s) on the current network structure. The set of candidate neighboring structures is called the *neighborhood* in the sequel. A classical neighborhood is composed of single arc

additions, deletions, and reversals. Using *larger* neighborhoods efficiently allows to find better local optima, and thus better network structures.

(Moore and Wong, 2003) proposed an optimal reinsertion of a target node by removing all its edges and reinserting it optimally. (Campos et al., 2002) explored a new reversal operator which deletes all the parents of both nodes if they share some parent, inverts the arc, and adds any subset of the union set of the old parents for each node. However these approaches are limited to small problems only. (Holland et al., 2008) used a restricted form of look ahead called LAGD, combining several operations in a single move. In this paper, we follow a similar approach by focusing our local operations on a target node and combining several operations to improve the score and restore the global acyclicity constraint of Bayes nets. By doing so, we are able to exploit large neighborhoods efficiently. Other approaches use a compact representation of a set of network structures like *Greedy Equivalence Search* (GES) (Chickering, 2002; Nielsen et al., 2003) which considers Markov-equivalent structures.

In Section 2, we give Bayesian network background. Next, we define a specific stochastic greedy search algorithm and introduce the new local move operators in Section 3. We report experimental results in Section 4 and conclude.

## 2 Bayesian network structure learning

A Bayesian network (Koller and Friedman, 2009) denoted by  $B = (G, \mathbf{P}_G)$  is composed of a directed acyclic graph (DAG)  $G = (\mathbf{X}, \mathbf{E})$  with nodes representing  $p$  random discrete variables  $\mathbf{X} = \{X_1, \dots, X_p\}$ , linked by a set of directed edges or *arcs*  $\mathbf{E}^1$ , and a set of conditional probability distributions  $\mathbf{P}_G = \{P_1, \dots, P_p\}$  defined by the topology of the graph:  $P_i = \mathbb{P}(X_i | Pa(X_i))$  where  $Pa(X_i) = \{X_j \in \mathbf{X} | (X_j \rightarrow X_i) \in \mathbf{E}\}$  is the set of parent nodes of  $X_i$  in  $G$ . A Bayesian network  $B$  represents a joint probability distribution on  $\mathbf{X}$  such that:  $\mathbb{P}(\mathbf{X}) = \prod_{i=1}^p \mathbb{P}(X_i | Pa(X_i))$ .

Conditional probability distributions  $\mathbf{P}_G$  are determined by a set of parameters. Given the network structure  $G$ , and the fully observed data  $D$ , parameters can be estimated by simple counting, following the maximum likelihood principle.

Learning the structure of a Bayesian network consists in finding a DAG  $G$  maximizing  $\mathbb{P}(G|D)$ . We have  $\mathbb{P}(G|D) \propto \mathbb{P}(D|G)\mathbb{P}(G)$ . Under specific assumptions, the *marginal loglikelihood*  $\log(\mathbb{P}(D|G))$  can be expressed as a decomposable scoring function  $f$  (e.g. BDeu score (Heckerman et al., 1995)):

$$f(D, G) = \sum_{i=1}^p f_{X_i}(D, G) = \sum_{i=1}^p f_{X_i}(D, Pa(X_i)) \quad (1)$$

A set of Bayesian networks are Markov-equivalent if they imply exactly the same set or *map* of independence constraints among variables<sup>2</sup>. Next, we describe a novel greedy search method maximizing  $f$  in the space of DAGs.

## 3 Stochastic Greedy Search

We define the Stochastic Greedy Search (SGS) algorithm for structural learning of Bayesian networks. It collects the best DAG found by  $r$  randomized hill-climbing algorithms. Stochasticity comes from two random draws. The first one, often considered, is the initial structure used for each restart. The second, more original, is when both edge orientations lead to Markov equivalent DAGs. The DAG is randomly drawn among the best Markov equivalent neighbors of the current DAG  $G$  at each step of the hill-climbing algorithm. The neighborhood of  $G$

is composed of the usual operations on DAGs: arc addition (ADD), arc deletion (DELETE) and arc reversal (REVERSE). This neighborhood is denoted  $\mathcal{N}^{ADR}$  in the sequel. Only operations which do not create cycles are considered. In the next subsections, we are going to extend this set of operations.

**Proposition 1.** (Gámez et al., 2011) *Let  $D$  be a dataset of  $n$  records that are identically and independently sampled from some distribution  $\mathbb{P}(\cdot)$ . Let  $f$  be a locally consistent scoring function. The hill-climbing algorithm in SGS returns a minimal independence map of  $\mathbb{P}(\cdot)$  as sample size  $n$  grows large.*

Recall that BDeu score is locally consistent (Chickering, 2002). The local consistency property ensures adding any arc that eliminates an independence constraint that does not hold in the generative distribution  $\mathbb{P}(\cdot)$  increases the score. Conversely, deleting any arc that results in a new independence constraint that holds in  $\mathbb{P}(\cdot)$  also increases the score.

The main interest of our randomization approach is to simulate a search in the space of score-equivalent networks. Each greedy search moves from a DAG instance randomly-selected from a Markov-equivalence class  $\mathcal{E}(G)$  to another DAG randomly-selected from an adjacent<sup>3</sup> Markov-equivalence class  $\mathcal{E}(G')$  thanks to our random selection among the best neighbors. It results in a stronger property:

**Proposition 2.** (Chickering, 2002) *Let  $D$  be a dataset of  $n$  iid fully observed samples of some faithful distribution  $\mathbb{P}(\cdot)$ . Let  $f$  be locally consistent. SGS returns a perfect map of  $\mathbb{P}(\cdot)$  as both the sample size  $n$  and the number of restarts  $r$  grow large.*

Recall a faithful distribution admits a unique perfect map corresponding to the optimal structure. Compared to the GES algorithm (Chickering, 2002), which offers the same optimality guarantee within a two-phase greedy search, SGS chooses the orientation of some *compelled arcs*<sup>4</sup> of the *true* DAG at random, whereas GES waits while no *v-structures* impose orientation constraints. See an example in Figure 1.

<sup>3</sup>Two equivalence classes  $\mathcal{E}(G)$  and  $\mathcal{E}(G')$  are adjacent iff  $G$  is an I-map of  $G'$  or vice-versa and the number of edges in the graphs  $G$  and  $G'$  differs by one.

<sup>4</sup>An arc  $X \rightarrow Y$  in  $G$  is *compelled* if that arc exists in every DAG of  $\mathcal{E}(G)$ , otherwise it is said *reversible*.

<sup>1</sup>We use  $G = \mathbf{E}$  when the set of nodes is implicit.

<sup>2</sup>BDeu give the same score for Markov-equivalent DAGs.

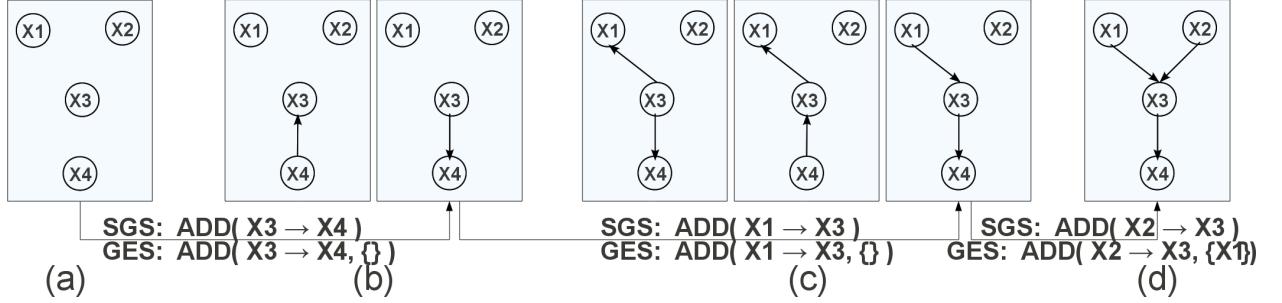


Figure 1: Four adjacent Markov-equivalence classes found by GES during its first phase of edge and v-structure insertions. (a) GES and SGS start from the empty graph . (d) The true DAG is found after three moves. The orientation of  $X_3 \rightarrow X_4$  and  $X_1 \rightarrow X_3$  edges are chosen at random by SGS, whereas GES waits until its third move to decide on edge orientations based on DAG score comparisons (enforcing the v-structure  $X_1 \rightarrow X_3 \leftarrow X_2$  as stated by the extra ADD parameter  $\{X_1\}$ , and forbidding  $X_1 \rightarrow X_3 \leftarrow X_4$  in its second move).

We observed in the experiments that a small number of restarts  $r$  allows to find DAGs with better scores than GES, especially when the sample size  $n$  is limited, in this case GES found a local optimum and SGS is able to find other better local optima thanks to randomization. This was also observed in (Nielsen et al., 2003).

When the sample size is small the learning problem becomes more difficult: the empirical distribution may be far from a perfect map resulting in many local optima and the scoring function is no more consistent, *i.e.* the likelihood does not dominate the penalty term on the complexity of the structure which is a non additive function of the parent variable domain sizes (Chickering, 2002). In this complex situation, we propose a new operator to escape from some local optima.

### 3.1 SWAP operator

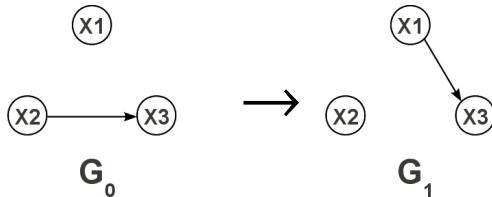


Figure 2: The operator SWAP( $X_2|X_1 \rightarrow X_3$ ) applied to a 3-variable problem.

Consider the 3-variable example in Figure 2 with observed data  $D$ , scoring function  $f$ , and initial DAG  $G_0 = \{X_2 \rightarrow X_3\}$ . Let assume  $f(D, \{X_1 \rightarrow$

$X_3\}) > f(D, \{X_2 \rightarrow X_3\}) > f(D, \{X_1 \rightarrow X_3, X_2 \rightarrow X_3\}) > f(D, \{X_3 \rightarrow X_1, X_2 \rightarrow X_3\}) > f(D, \{X_2 \rightarrow X_1, X_2 \rightarrow X_3\}) > f(D, \emptyset)$ . Then  $G_0$  is a local minimum for the classical neighborhood  $\mathcal{N}^{ADR}$ . Our new operator, denoted SWAP( $X|Y \rightarrow Z$ ), consists in changing one parent  $X$  to another parent  $Y$  for one target node  $Z$ . This is equivalent to a simultaneous pair of ADD and DELETE operators restricted to the same target node. In our example, applying SWAP( $X_2|X_1 \rightarrow X_3$ ) corresponds to DELETE( $X_2 \rightarrow X_3$ ),ADD( $X_1 \rightarrow X_3$ ), resulting in the better DAG  $G_1 = \{X_1 \rightarrow X_3\}$  as shown in Figure 2. The extended neighborhood using the 4 operators is denoted  $\mathcal{N}^{ADRS}$  and SGS using  $\mathcal{N}^{ADRS}$  (respectively  $\mathcal{N}^{ADR}$ ) is denoted SGS<sup>2</sup> (SGS with Swap) (resp. SGS<sup>1</sup>) in the sequel.

Let  $p$  be the number of variables in the DAG and  $k$  be the maximum number of parents per node. Assuming a sparse graph,  $p \gg k$ , the number of SWAP operations is larger than for classical operators but it remains bounded by  $O(kp^2)$ , the complexity of  $\mathcal{N}^{ADRS}$  is therefore in  $O(kp^2)$ , whereas it is in  $O(p^2)$  for  $\mathcal{N}^{ADR}$ . Other approaches using larger neighborhoods such as *h-look ahead in l good directions* (LAGD) has a worst-case complexity in  $O(l^{h-1}p^2)$  (Holland et al., 2008), optimal reinsertion is in  $O(2^k p^{k+1})$  (Moore and Wong, 2003), and modified reversal in  $O(2^{2k} p^2)$  (Campos et al., 2002).

Another source of suboptimality comes from the global acyclicity constraint of Bayesian networks.

### 3.2 Breaking cycles by successive deletions and swaps

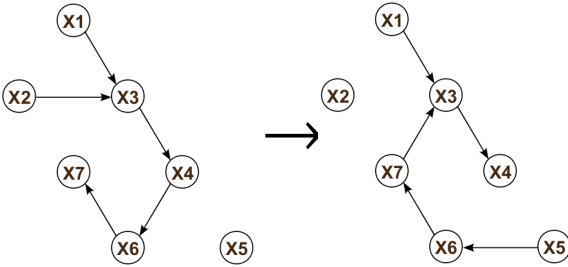


Figure 3: Applying an extended  $\text{SWAP}^*$  operation breaking a cycle by an additional  $\text{SWAP}$  operation:  $\text{SWAP}^*(X2|X7 \rightarrow X3) = \{\text{SWAP}(X2|X7 \rightarrow X3), \text{SWAP}(X4|X5 \rightarrow X6)\}$ .

Consider the 7-variable DAG example in Figure 3. Swapping the parent  $X2$  of  $X3$  by  $X7$  in DAG  $G$  (Fig. 3.left) introduces a directed cycle  $\{X7 \rightarrow X3, X3 \rightarrow X4, X4 \rightarrow X6, X6 \rightarrow X7\}$  and is therefore forbidden in our  $N^{ADRS}$  neighborhood. However it may correspond to a large *local* score improvement with respect to variable  $X3$ . Let us denote this improvement by  $\Delta_{X3}(G, \text{SWAP}(X2|X7 \rightarrow X3)) = f_{X3}(D, G') - f_{X3}(D, G)$  with  $G'$  obtained by applying the  $\text{SWAP}$  operation on  $G$  ( $G'$  is not a valid DAG), and  $D$  and  $f$  being the sample and scoring function. Our idea is to heuristically guide the search for a second (or more) local operator to be applied on  $G'$  in order to restore graph acyclicity ( $G'$  becomes valid) and such that the *true* score of the final DAG is greater than the score of the original one. In Figure 3, it is obtained by applying a second  $\text{SWAP}$ .

For that purpose, we define an extended  $\text{SWAP}$  operator, denoted  $\text{SWAP}^*$ , able to break all directed cycles by performing a succession of deletion or swap operations. It can be seen as a kind of greedy descent search in the space of directed cyclic graphs, trying to remove the less important arcs or to swap them in order to compensate for their loss, until a better valid DAG is found. We use local score changes to guide the search:  $\Delta_{Xi}(G, OP) = f_{Xi}(D, G') - f_{Xi}(D, G)$ , with  $G'$  the result of applying the local move operator  $OP \in \{\text{DELETE}, \text{SWAP}\}$  to  $G$ . A negative sum of local changes aborts the search. Recall that finding a minimum number of arc deletions in order to restore acyclicity is NP-

**Algorithm 1:**  $\text{SWAP}^*(X|Y \rightarrow Z)$  operator.

---

|               |                                                                                            |
|---------------|--------------------------------------------------------------------------------------------|
| <b>Input</b>  | : operation $X Y \rightarrow Z$ , sample $D$ , score $f$ , DAG $G(\mathbf{X}, \mathbf{E})$ |
| <b>Output</b> | : a set of local operations $\mathbf{L}$                                                   |

```

 $L \leftarrow \emptyset$ /* Initialize output operations to the empty set */;
 $X' \leftarrow X$ /* Candidate parent set for future swaps */;
 $G' \leftarrow G$ /* Copy of input DAG */;
1 $\Delta = \Delta_Z(G', \text{SWAP}(X|Y \rightarrow Z))$ /* Putative score improvement */;
if $\Delta > 0$ then
 $L \leftarrow L \cup \{\text{SWAP}(X|Y \rightarrow Z)\}$;
 Apply $\text{SWAP}(X|Y \rightarrow Z)$ to G' ;
 /* Repeat deletion or swap operations until no more cycles */ while $\Delta > 0 \wedge (\mathbf{C} \leftarrow \text{NextCycle}(G')) \neq \emptyset$
 do
 2 $X' \leftarrow X' \setminus \text{nodes}(\mathbf{C})$;
 /* Choose the best deletion to break cycle \mathbf{C} */;
 $(U^* \rightarrow W^*) \leftarrow \arg\max_{(U \rightarrow W) \in \mathbf{C} \setminus \{Y \rightarrow Z\}} \Delta_W(G', \text{DELETE}(U \rightarrow W))$;
 /* Test if the sum of local score changes is positive */;
 if $\Delta + \Delta_{W^*}(G', \text{DELETE}(U^* \rightarrow W^*)) > 0$ then
 $L \leftarrow L \cup \{\text{DELETE}(U^* \rightarrow W^*)\}$;
 $\Delta \leftarrow \Delta + \Delta_{W^*}(G', \text{DELETE}(U^* \rightarrow W^*))$;
 Apply $\text{DELETE}(U^* \rightarrow W^*)$ to G' ;
 else
 3 $/ * \text{Choose the best swap to get a positive change} */$;
 $(U^*|V^* \rightarrow W^*) \leftarrow \arg\max_{(U \rightarrow W) \in \mathbf{C}, V \in X} \Delta_W(G', \text{SWAP}(U|V \rightarrow W))$;
 $\Delta \leftarrow \Delta + \Delta_{W^*}(G', \text{SWAP}(U^*|V^* \rightarrow W^*))$;
 if $\Delta > 0$ then
 $L \leftarrow L \cup \{\text{SWAP}(U^*|V^* \rightarrow W^*)\}$;
 Apply $\text{SWAP}(U^*|V^* \rightarrow W^*)$ to G' ;
 else
 4 $L \leftarrow \emptyset$ /* Abort all local operations */;
 return L ;
```

---

hard. We use a greedy approach instead. The pseudo-code of  $\text{SWAP}^*$  is given in Algorithm 1. The local score improvement of the initial  $\text{SWAP}$  operation is evaluated at line 1. It corresponds to a putative gain on the current score. If it is positive then this operation is applied to a copy of the input DAG  $G$ , checking next if it creates some directed cycles. Each cycle is retrieved by the  $\text{NextCycle}$  function and the algorithm searches for an arc deletion in this cycle with minimum deterioration of the local score at line 4. If the combined local score change of the  $\text{SWAP}$  and  $\text{DELETE}$  operations is positive then it applies the selected arc deletion and continues to test if there are no more directed cycles at line 2. If the combined local score change is negative then it tries to swap an arc of the cycle such

that the combined local score change is maximized (line 5) and positive. If it fails to find such an operation then it stops breaking cycles and returns an empty operation set. Finally if it succeeds, breaking all cycles, then it returns a feasible set of SWAP and DELETE operations resulting into a new valid DAG  $G'$  with a better score than  $G$ . The true score improvement is equal to  $\Delta$ .

We used a restricted list of alternative candidate parent nodes  $\mathbf{X}'$  at line 5 to avoid that new arcs created by swap operations were swapped again, which guarantees that algorithm 1 terminates.

We apply the same approach to extend the operator ADD\*, we note that REVERSE\* is unnecessary since  $\text{REVERSE}^*(X \rightarrow Y) = \text{ADD}^*(Y \rightarrow X)$ . The resulting neighborhood exploiting these extended operators is denoted  $N^{ADD^*}$  and SGS using this neighborhood is denoted SGS<sup>3</sup> (Stochastic Greedy Search with Successive Swaps) in the experiments.

## 4 Experimental Results

In this section, we describe a set of experiments aimed at testing the performance of SGS<sup>i</sup> algorithms compared with state-of-the-art Bayesian network structure learning algorithms on standard Bayesian nets and challenging gene regulatory nets.

### 4.1 Results on Standard Bayesian Networks

We used four gold-standard networks from the Bayesian Network Repository<sup>5</sup>: A , I , H and P networks. The number of nodes varies from 27 to 441 with a connectivity value around 1.5. 100 samples of size  $n = 500$  and  $n = 5,000$  were generated for each network using Causal Explorer (Aliferis et al., 2003).

We compared SGS<sup>i</sup> algorithms with LAGD (Holland et al., 2008), available in the WEKA software (Hall et al., 2009) and GES (Chickering, 2002) implemented in Tetrad 4.4.0 (Scheines et al., 1998). LAGD and GES were shown to outperform or to have similar performance to several recent algorithms in (Salehi and Gras, 2009; Alonso-Barba et al., 2011). Recall that SGS<sup>1</sup> is similar to a repeated randomized orientations hill-climbing, SGS<sup>2</sup> uses the SWAP operator, and SGS<sup>3</sup> breaks cycles

<sup>5</sup><http://www.cs.huji.ac.il/site//labs/compbio/repository/>

by successive DELETE and SWAP operators. Experiments were performed on a 3 GHz Intel Core2 computer with 4 GB running Linux 2.6.

We fixed the maximum number of parents per node at  $k = 5$  for SGS<sup>i</sup> and LAGD. LAGD exploits a  $h = 2$ -look ahead in  $l = 5$  good directions. GES was restricted on the number of adjacent nodes:  $d = 7$  for Hailfinder and  $d = 10$  for Pigs network as done in (Alonso-Barba et al., 2011). All methods were initialized by an empty graph and optimized the BDeu score with *equivalent sample size*  $\alpha = 1$  and no prior on the network structures. For each sample, we recorded the best score obtained by GES, and by  $r = 10$  randomized greedy searches for SGS<sup>i</sup> as for LAGD<sup>6</sup>.

Table 1: Wilcoxon test comparing pairs of algorithms (familywise error rate = 5%). For Method1 versus Method2, + means that Method1 is significantly better than Method2, – means that Method1 is significantly worse than Method2 and ~ means there is no significant result

| Sample size              | A   |       | I   |       |
|--------------------------|-----|-------|-----|-------|
|                          | 500 | 5,000 | 500 | 5,000 |
| SGS <sup>3</sup> vs GES  | +   | +     | +   | +     |
| SGS <sup>3</sup> vs LAGD | +   | +     | +   | +     |
| LAGD vs GES              | +   | ~     | +   | +     |
| H                        |     | P     |     |       |
| SGS <sup>3</sup> vs GES  | +   | +     | +   | -     |
| SGS <sup>3</sup> vs LAGD | ~   | +     | n/a | n/a   |
| LAGD vs GES              | +   | +     | n/a | n/a   |

In order to test the statistical significance of the difference in BDeu score between two methods, we applied a non-parametric paired test, the Wilcoxon signed-rank test (Wilcoxon, 1945). Table 1 presents the test results for SGS<sup>3</sup> (which outperformed SGS<sup>1</sup> and SGS<sup>2</sup>), LAGD and GES by using an unilateral alternative (no difference versus better) and a familywise error rate of 5%.

SGS<sup>3</sup> was the best method for the four networks, except for Pigs network with  $n = 5,000$  which is more accurately estimated by GES. We conjecture that in this case, GES was closed to its asymptotic optimal behavior, which may be due to the structure of Pigs network with small nodes in-degree. LAGD

<sup>6</sup>We randomly permute the input variables at each run.

failed on the Pigs network due to the large number of variables  $p = 441$  that makes the exploration of 2-look ahead neighborhoods infeasible in a reasonable time. GES was the worst method here (except for Pigs) due to limited sample sizes. Results not shown here indicate that SGS<sup>2</sup> improved over SGS<sup>1</sup> and reached the second position and SGS<sup>1</sup> outperformed LAGD which can be explained by a better randomization in our implementation.

Although the algorithms are designed to maximize a (BDeu) score, we generally look for a network structure as close as possible to the true one. We report in Table 2 the means over 100 datasets (rounded values to the nearest integer) of the missing and spurious edges without taking into account the edge orientations. The *structural Hamming distance* (SHD) is the sum of the above values. SGS<sup>3</sup> (resp. GES) got the best SHD in 4 (resp. 5) configurations and outperformed LAGD (which performed as SGS<sup>3</sup> in 1 configuration). GES performed extremely well on the Pigs network, finding the true network with 5,000 samples, whereas SGS<sup>3</sup> learned too many edges but recovered all the true edges (even with  $n = 500$ ). The spurious edges learned by SGS<sup>3</sup> are exclusively due to random orientations of compelled arcs in v-structures (see Figure 1). Assuming  $X_1 \rightarrow X_3 \leftarrow X_2$  in the true network (v-structures are very frequent in the Pigs network) and a large sample size, if during its greedy search SGS<sup>3</sup> adds first  $X_1 \leftarrow X_3$  and  $X_3 \rightarrow X_2$ , it will add next a *covering edge*  $X_1 \rightarrow X_2$  or  $X_1 \leftarrow X_2$  in order to find a minimal I-map (see Proposition 1). These *covered v-structures* can be found in post-processing by selecting for each one the best configuration among the 3 possible v-structures.

## 4.2 Detailed analysis on the Alarm network

We further analyzed the impact on performances of the number of restarts  $r$  and the initial graph used by SGS <sup>$i$</sup>  algorithms on the Alarm network with a sample size  $n = 500$ . Figure 4 reports averaged BDeu scores on 30 Alarm samples. The 30 initial random graphs, the same set used by all the SGS <sup>$i$</sup>  algorithms, are composed of 71 arcs with at most two parents per node<sup>7</sup>. All SGS <sup>$i$</sup>  methods reached a better BDeu score than GES in this small sample size

<sup>7</sup>For each node, we randomly select two parents and remove a parent if it creates a directed cycle.

Table 2: Number of spurious edges (+) and missing edges (-) to sum for the structural Hamming distance.

| Sample size        | A         |           | I         |          |
|--------------------|-----------|-----------|-----------|----------|
|                    | 500       | 5,000     | 500       | 5,000    |
| SGS <sup>3</sup> + | <b>8</b>  | 6         | <b>4</b>  | <b>2</b> |
| SGS <sup>3</sup> - | <b>3</b>  | 2         | <b>20</b> | <b>8</b> |
| LAGD+              | 11        | 8         | <b>4</b>  | 5        |
| LAGD-              | 4         | 2         | <b>20</b> | 11       |
| GES+               | <b>6</b>  | <b>4</b>  | 2         | 3        |
| GES-               | <b>5</b>  | <b>2</b>  | 23        | 12       |
|                    | H         |           | P         |          |
| SGS <sup>3</sup> + | 17        | <b>16</b> | 32        | 41       |
| SGS <sup>3</sup> - | 24        | <b>13</b> | 0         | 0        |
| LAGD+              | 21        | 20        | n/a       | n/a      |
| LAGD-              | 26        | 19        | n/a       | n/a      |
| GES+               | <b>15</b> | 11        | <b>2</b>  | <b>0</b> |
| GES-               | <b>24</b> | 22        | <b>7</b>  | <b>0</b> |

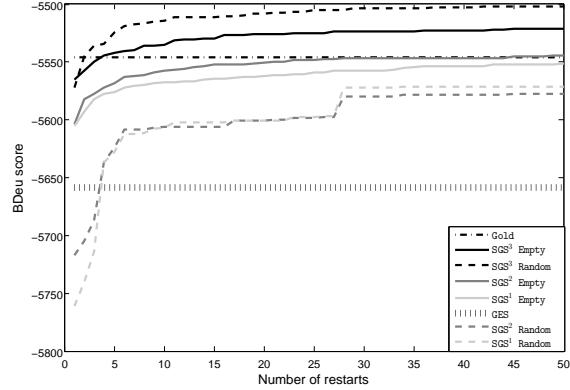


Figure 4: Best BDeu scores, averaged on 30 Alarm samples ( $n = 500$ ), found by SGS <sup>$i$</sup>  algorithms as the number of restarts  $r$  increases and starting either from an empty (solid line) or a random graph (dashed line). Results of GES (dotted line) and BDeu score of the true network *Gold* (dash-dotted line) are also given. Methods are sorted by decreasing BDeu score at  $r = 1$ .

situation by the use of less than  $r = 4$  restarts. SGS <sup>$i$</sup>  methods converged quite rapidly as  $r$  increases. For a fixed  $r$ , SGS<sup>1</sup> was twice faster than SGS<sup>3</sup>. Only SGS<sup>3</sup> found a better score than the true network. Initial random graphs were counter-productive for all the methods, except for SGS<sup>3</sup>. It shows that start-

ing from a random graph is useful if the available operators are able to move efficiently in the search space. On the contrary, using randomness to select among the best neighbors was always beneficial.

### 4.3 Results on Gene Regulatory Networks

Gene regulatory network reconstruction from gene expression data using Bayesian network structure learning was first proposed in (Friedman et al., 2000). We used simulated expression datasets of the DREAM5 Systems Genetics Challenge A (de la Fuente, 2010). Genetics data were not used as they require additional modelling to be taken into account, see *e.g.* (Vignes et al., 2011). Expression data were generated using the SysGenSIM generator (Pinna et al., 2011) based on ordinary differential equation simulation. Five datasets are available for three different sample sizes ( $n = 100, 300$ , and  $999$ ). The 15 datasets were obtained from *different* known gene networks composed of 1,000 variables and containing directed cycles. For each sample size, the five network structures contain a different number of edges varying from  $\approx 2,000$  (*Net1*) to more than 5,000 (*Net5*). We discretized gene expression levels into 2 to 4 states using an adaptive method based on an adapted  $k$ -means algorithm and the more general framework of Gaussian mixture models as described in (Vignes et al., 2011).

With such large networks, we had to adapt the learning procedure of  $SGS^i$  algorithms<sup>8</sup>. We decided to restrict their lists of candidate parents as done in (Goldenberg and Moore, 2004): we selected for each variable  $X$  the set of parents  $S$  such that each element  $Y$  of  $S$  improves the local BDeu score when it is considered as a unique parent compared to the orphan situation ( $f_X(D, \{Y \rightarrow X\}) > f_X(D, \emptyset)$ ). This filtering process was done before the search. In these experiments,  $SGS^i$  algorithms have a maximum number of parents per node fixed at  $k = 5$  and use  $r = 10$  restarts. Instead of LAGD (which was too slow), we used MMHC (Tsamardinos et al., 2006) having two steps similar to  $SGS^i$  but using mutual information measures. We recorded the best BDeu score of 10 runs for MMHC, by randomly permuting the input variables at each run. All the methods started from an empty graph and optimized

<sup>8</sup>GES managed in ~1-hour CPU time each network thanks to its better implementation of caching and heap data structure.

the BDeu score with  $\alpha = 1$  and no prior on the network structures.

Table 3: Wilcoxon test (error rate = 5%) for different gene network sample sizes

|                 | 100 | 300 | 999 |
|-----------------|-----|-----|-----|
| $SGS^3$ vs MMHC | +   | +   | +   |
| $SGS^3$ vs GES  | +   | +   | +   |
| MMHC vs GES     | -   | ~   | +   |

As there were no replicated samples of the same network, we performed the Wilcoxon test on pooled groups for each sample size. We applied a pairwise type I error of 5% and we did not try to correct for multiple comparisons, see Table 3. However, it is worth noting  $SGS^3$  was always the best method, increasing the BDeu score by about 2% in average.

Surprisingly, GES appeared to be better on smaller sample sizes compared to MMHC. MMHC was penalized by its filtering process, especially on the smallest sample size, whereas GES had no restrictions on the candidate parent sets.

In these experiments, the structural Hamming distance (SHD) was not informative due to the poor results reached by all tested algorithms for such large networks, even the empty structure appears better. For this reason, we computed the Euclidean distance to the origin ( $\sqrt{precision^2 + recall^2}$ ). Contrary to SHD, a high distance indicates a better structural quality. We observed in Table 4 contrasted performances between the tested methods depending on the sample size: for  $n=100$ , MMHC got the best results, for  $n = 300$ , it was GES, and finally  $SGS^3$  performed the best for the largest sample size. Better BDeu scores are not always synonymous with a better structural quality, the limited sample size in addition to the non faithfulness of the data could explain this behavior.

Table 4: Euclidean distances to the origin of the (precision, recall) values. Means of the 5 networks for each sample size.

| $n$ | $SGS^3$      | MMHC         | GES          |
|-----|--------------|--------------|--------------|
| 100 | 0.201        | <b>0.258</b> | 0.224        |
| 300 | 0.442        | 0.437        | <b>0.456</b> |
| 999 | <b>0.514</b> | 0.482        | 0.500        |

## 5 Conclusion

We presented in this paper new greedy search algorithms called SGS<sup>i</sup> exploiting stochasticity from two random draws. We have developed a new local move operator called SWAP and extended versions for ADD and SWAP operators to overcome frequent limitations of local search methods which are local maxima and cyclic situations. We compared SGS<sup>3</sup> using SWAP and extended operators to state-of-the-art methods and we obtained significant BDeu improvements on classical benchmarks and also simulated gene regulatory network datasets when the sample size is small. The complexity of SGS<sup>3</sup> stays moderate with sparse networks.

In the future, we would like to test our operators with other local search methods like tabu search.

**Acknowledgements** We would like to thank the anonymous reviewers for their helpful comments.

## References

- C. Aliferis, I. Tsamardinos, A. Statnikov, and L. Brown. 2003. Causal Explorer: A Probabilistic Network Learning Toolkit for Biomedical Discovery. In *Proc. of METMBS'2003*, Las Vegas, Nevada, USA.
- J. Alonso-Barba, L. de la Ossa, and J. Puerta. 2011. Structural learning of bayesian networks using local algorithms based on the space of orderings. *Soft Comput.*, pages 1881–1895.
- L.M. de Campos, J.M. Fernandez-Luna, and J.M. Puerta. 2002. Local Search Methods for Learning Bayesian Networks Using a Modified Neighborhood in the Space of DAGs. In *LNCS (2527)*, pages 182–192.
- D. Chickering and D. Heckermann. 1996. Learning bayesian networks is NP-complete. In *learning from data: Al and Statistics*.
- D. Chickering. 2002. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.
- R. Daly, Q. Shen, and S. Aitken. 2011. Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review*, 26(2):99–157.
- A. de la Fuente. 2010. The DREAM5 Systems Genetics Challenges. <http://wiki.c2b2.columbia.edu/dream/index.php/D5c3>.
- N. Friedman, M. Linial, I. Nachman, and D. Pe'er. 2000. Using bayesian networks to analyse expression data. *Journal of computational biology*, 7(3/4):601–620.
- J. Gámez, J. Mateo, and J. Puerta. 2011. Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Min. Knowl. Discov.*, 22:106–148.
- A. Goldenberg and A. Moore. 2004. Tractable learning of large Bayes net structures from sparse data. In *Proc. of ICML'04*, pages 44–51.
- M. Hall, F. Eibe, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. 2009. The WEKA Data Mining Software. *SIGKDD Explorations*, 11.
- D. Heckerman, D. Geiger, and D. Chickering. 1995. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. In *Machine Learning*, volume 20, pages 197–243.
- A. Holland, M. Fathi, M. Abramovici, and M. Neubach. 2008. Competing fusion for bayesian applications. In *Proc. of IPMU 2008*, pages 378–385.
- D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.
- A. Moore and W.K. Wong. 2003. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In *Proc. of ICML '03*, pages 552–559.
- J. Nielsen, T. Kocka, and J. Pefia. 2003. On Local Optima in Learning Bayesian Networks. In *Proc. of UAI-03*, pages 435–442.
- A. Pinna, N. Soranzo, I. Hoeschele, and A. de la Fuente. 2011. Simulating system genetics data with SysGen-SIM. *Bioinformatics*, 27:2459–2462.
- E. Salehi and R. Gras. 2009. An empirical comparison of the efficiency of several local search heuristics algorithms for Bayesian network structure learning. In *Proc. of LION 3*, Trento, Italy.
- R. Scheines, P. Spirtes, C. Glymour, C. Meek, and T. Richardson. 1998. The TETRAD Project: Constraint Based Aids to Causal Model Specification. *Multivariate Behavioral Research*, 33(1):65–117.
- I. Tsamardinos, L. Brown, and C. Aliferis. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.*, 65:31–78.
- M. Vignes, J. Vandel, D. Allouche, N. Ramadan, C. Cierco, T. Schiex, B. Mangin, and S. de Givry. 2011. Gene Regulatory Network Reconstruction Using Bayesian Networks, the Dantzig Selector, the Lasso and Their Meta-Analysis. *PLoS ONE*, 6(12).
- F. Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1:80–83.

# Flood Damage and Influencing Factors: A Bayesian Network Perspective

Kristin Vogel <sup>1</sup>

kvog@geo.uni-potsdam.de

Carsten Riggelsen <sup>1</sup>

riggelsen@geo.uni-potsdam.de

Bruno Merz <sup>2</sup>

bmerz@gfz-potsdam.de

Heidi Kreibich <sup>2</sup>

kreib@gfz-potsdam.de

Frank Scherbaum <sup>1</sup>

fs@geo.uni-potsdam.de

<sup>1</sup> Institute of Earth and Environmental Science, University of Potsdam, Germany

<sup>2</sup> GFZ – GeoForschungszentrum Potsdam, Germany

## Abstract

Classical approaches for flood risk assessment relate flood damage for a certain class of objects to the inundation depth, while other characteristics of the flooding situation and the flooded object are widely ignored. Observations on several discrete and continuous variables collected after the 2002 and 2005/2006 floods in the Elbe and Danube catchments in Germany offer a unique data mining opportunity in terms of learning a Bayesian Network. We take an entirely data-driven stance opting not to discretize continuous variables in advance; rather, we cast the problem in Bayesian framework, and consider the *maximum a posteriori* of the joint distribution of the triple, network structure, parameters and discretization, as the outcome of the analysis. Moreover, motivated by the work of Merz et al. (2010), who point out the need of an improved flood damage assessment, we re-define the discretization of the target variable, flood loss, once the network has been learned. Its domain is split into a large number of intervals and the associated parameters are estimated using a Gaussian kernel density estimator. Although the prediction of the relative flood loss is comparable to state-of-the-art methods, our approach benefits from capturing the joint distribution of all factors influencing flood loss.

## 1 Introduction

Graphical models have in recent years successfully been employed in earth sciences, giving rise to a wide range of applications, including Tsunami Early Warning, e.g. (Blaser et al., 2011), Probabilistic Seismic Hazard Analysis, e.g. (Kuehn et al., 2011), and Automatic detection and classification of seismic signals, e.g.

(Riggelsen et al., 2007). In this paper we embark on another problem: flood damage assessment of residential buildings. Typically, the damage to flooded objects is estimated by stage-damage functions which relate the relative or absolute damage for a certain class of objects to the water stage or inundation depth (Merz et al., 2010). Other characteristics of the flooding

situation and of the flooded object are rarely taken into account, although it is clear that flood damage is influenced by a variety of factors such as inundation duration, contamination of flood water, or quality of external response in a flood situation. The single and joint effects of these parameters on the degree of damage are largely unknown and widely neglected in damage assessments. Moreover, the intrinsic uncertainties associated with these factors are largely ignored. *Bayesian Networks* (BN) pose an interesting formalism for capturing the interdependencies and the intrinsic uncertainty involved in flood risk assessment.

The paper is organized as follows: After giving a description of the data set in Section 2 and a brief introduction into learning BNs in Section 3, we show in Section 4, how we automatically discretize continuous variables, based on the observed data set, using a maximum a posteriori (MAP) score to search simultaneously for the best network structure, parameters and discretization. In Section 4.1 we point out, how we employ the learned BN to estimate the flood loss, using a very fine discretization of the target variable to allow a precise approximation of its continuous conditional distribution functions. Finally, the results are shown in Section 5 and we conclude in Section 6.

## 2 Variable Definitions and Dataset

We take a data mining perspective and aim for learning a BN from observational data. The observations are collected after the 2002 and 2005/2006 floods in the Elbe and Danube catchments in Germany. Results of computer-aided telephone interviews with 1135 flood affected households yield i.i.d. data,  $\mathbf{d} = \cup_k \{\mathbf{x}^{(k)}\}$ . Topics relate to various flood parameters (e.g. contamination, water depth), building and household characteristics, precautionary measures, and flood damage to buildings and contents. The raw data were supplemented by estimates of return periods, building values, loss ratio, i.e. the relation between the building damage and the building value, and indicators for flow velocity, contamination, flood warning, emergency

| $X_i$                                 | Predictors                                                       | Scale and range                                                                                  |
|---------------------------------------|------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| <b>flood parameters</b>               |                                                                  |                                                                                                  |
| wst                                   | Water depth                                                      | C: 248 cm below ground to 670 cm above ground                                                    |
| d                                     | Inundation duration                                              | C: 1 to 1440 h                                                                                   |
| v                                     | Flow velocity indicator                                          | O: 0=still to 3=high velocity                                                                    |
| con                                   | Contamination indicator                                          | O: 0=no contamination to 6=heavy contamination                                                   |
| rp                                    | Return period                                                    | C: 1 to 848 yrs                                                                                  |
| <b>warning and emergency measures</b> |                                                                  |                                                                                                  |
| wt                                    | Early warning lead time                                          | C: 0 to 336 h                                                                                    |
| wq                                    | Quality of warning                                               | O: 1=receiver of warning knew exactly what to do to 6=receiver of warning had no idea what to do |
| ws                                    | Indicator of flood warning source                                | N: 0=no warning to 4=official warning through authorities                                        |
| wi                                    | Indicator of flood warning information                           | O: 0=no helpful information to 11=many helpful information                                       |
| wte                                   | Lead time period elapsed without using it for emergency measures | C: 0 to 335 h                                                                                    |
| em                                    | Emergency measures indicator                                     | O: 1=no measures undertaken to 17=many measures undertaken                                       |
| <b>precaution</b>                     |                                                                  |                                                                                                  |
| pre                                   | Precautionary measures indicator                                 | O: 0=no measures undertaken to 38=many, efficient measures undertaken                            |
| epre                                  | Perception of efficiency of private precaution                   | O: 1=very efficient to 6=not efficient at all                                                    |
| fe                                    | Flood experience indicator                                       | O: 0=no experience to 9=recent flood experience                                                  |
| kh                                    | Knowledge of flood hazard                                        | N (yes / no)                                                                                     |
| <b>building characteristics</b>       |                                                                  |                                                                                                  |
| bt                                    | Building type                                                    | N (1=multifamily house, 2=semi-detached house, 3=one-family house)                               |
| nfb                                   | Number of flats in building                                      | C: 1 to 45 flats                                                                                 |
| fsb                                   | Floor space of building                                          | C: 45 to 18000 m <sup>2</sup>                                                                    |
| bq                                    | Building quality                                                 | O: 1=very good to 6=very bad                                                                     |
| bv                                    | Building value                                                   | C: 92244 to 3718677                                                                              |
| <b>socio-economic factors</b>         |                                                                  |                                                                                                  |
| age                                   | Age of the interviewed person                                    | C: 16 to 95 yrs                                                                                  |
| hs                                    | Household size, i.e. number of persons                           | C: 1 to 20 people                                                                                |
| chi                                   | Number of children (< 14 years) in household                     | C: 0 to 6                                                                                        |
| eld                                   | Number of elderly persons (> 65 years) in household              | C: 0 to 4                                                                                        |
| own                                   | Ownership structure                                              | N (1=tenant; 2=owner of flat; 3=owner of building)                                               |
| inc                                   | Monthly net income in classes                                    | O: 11=below 500 to 16=3000 and more                                                              |
| socP                                  | Socioeconomic status according to Plapp (2003)                   | O: 3=very low socioeconomic status to 13=very high socioeconomic status                          |
| socS                                  | Socioeconomic status according to Schnell et al (1999)           | O: 9=very low socioeconomic status to 60=very high socioeconomic status                          |
| <b>flood loss</b>                     |                                                                  |                                                                                                  |
| rloss                                 | loss ratio of residential building                               | C: 0 = no damage to 1 = total damage                                                             |

Table 1: Description of the candidate predictors  
(C: continuous, O: ordinal, N: nominal).

measures, precautionary measures, flood experience and socioeconomic variables (Thieken et al., 2005; Elmer et al., 2010). Table 1 lists 28 candidate variables allocated to 5 domains and the predictand *rloss*, which is the direct damage to flooded residential buildings represented as relative value, i.e. fraction of the building value.

### 3 Bayesian Network Learning

Formally speaking a BN decomposes a joint probability distribution/density  $P(\mathbf{X})$  into a product of (local) conditional probability distributions/densities  $p(\cdot|\cdot)$  as  $P(\mathbf{X}) = \prod_i p(X_i|\mathbf{X}_{Pa(i)})$  according to a directed acyclic graph (DAG) with vertices  $X_i$  and directed edges from variables in the parent set  $\mathbf{X}_{Pa(i)}$  to  $X_i$ . In case  $\mathbf{X}$  is continuous and we do not know  $p(\cdot|\cdot)$  in advance, we may approximate the (local) conditional probability distributions by first discretizing the continuous variables and rely on contingency tables instead; the challenges that this involves is discussed in Section 4. For such a discrete BN we write  $P(\mathbf{X}|DAG, \boldsymbol{\theta}) = \prod_i \theta_{X_i|\mathbf{X}_{Pa(i)}}$ , where  $\boldsymbol{\theta}$  (the *parameters*) are conditional probabilities derived from contingency tables. For the rest of this section we assume all variables  $\mathbf{X}$  to be discrete.

BN model selection (learning the joint decomposition as well as the local conditional probabilities) is an exercise in traversing the space of BNs looking for the one which maximizes a given fitness score. As usual for model selection, regularization plays a role in this endeavour. We use the Bayesian BN MAP score (Riggelsen, 2008) shown to learn BN that are better than those derived via the marginal likelihood score (the BD-score). The BN is selected as the MAP of the joint posterior (here both DAG and parameter are being treated as true random variables)

$$P(DAG, \boldsymbol{\Theta}|\mathbf{d}) \propto P(\mathbf{d}|DAG, \boldsymbol{\Theta})P(\boldsymbol{\Theta}, DAG),$$

where the joint prior is a product, with  $P(\boldsymbol{\Theta}|DAG)$  defined to be a product Dirichlet distribution and  $P(DAG)$  defined to be uniform over DAGs (we may thus ignore this term

when doing MAP estimation). The simultaneous joint selection of DAG and parameters yields a DAG which equivalently can be found by maximizing the structure score

$$\begin{aligned} S(DAG|\mathbf{d}) &= \prod_{i, \mathbf{x}_{Pa(i)}, x_i} \hat{\theta}_{x_i|\mathbf{x}_{Pa(i)}}^{n(x_i, \mathbf{x}_{Pa(i)}) + \alpha(x_i, \mathbf{x}_{Pa(i)})} \\ &\times \underbrace{\prod_{i, \mathbf{x}_{Pa(i)}} \frac{\Gamma(\sum_{x_i} \alpha(x_i, \mathbf{x}_{Pa(i)}))}{\prod_{x_i} \Gamma(\alpha(x_i, \mathbf{x}_{Pa(i)}))}}_{\text{regularization}} \end{aligned}$$

where the statistics  $n(\cdot)$  are the counts of a particular configuration from the data/contingency table and  $\alpha(\cdot)$  are the hyper-parameters of the Dirichlet (restricted as to guarantee uniform DAG scoring equivalence; see (Riggelsen, 2008)). The BN parameter estimates required for computing the above score are also the BN parameters, and are given in closed form by

$$\hat{\theta}_{x_i|\mathbf{x}_{Pa(i)}} = \frac{n(x_i, \mathbf{x}_{Pa(i)}) + \alpha(x_i, \mathbf{x}_{Pa(i)})}{n(\mathbf{x}_{Pa(i)}) + \alpha(\mathbf{x}_{Pa(i)})}. \quad (1)$$

The BN is learned using a hill-climber approach in the space of DAGs based on the score given above where arc addition, removal and reversal are the basic operations. DAG equivalent classes are simulated using the Repeated Covered Arc Reversal operator (Castelo and Kocka, 2003).

### 4 Automatic Discretization

We want to adhere to an entirely data-driven approach for learning BNs and strive for making none or at least very weak assumption with regard to the functional form of the (local) conditional distributions,  $p(X_i|\mathbf{X}_{Pa(i)})$ . A sufficiently fine discretization of continuous variables, placing “counts” in contingency tables, enables us to approximate any other (local conditional) distribution, e.g., a Gaussian, but usually with a larger number of parameters.

To transform a continuous variable into a discrete one, the number of intervals and their boundaries have to be chosen carefully. A fine-grained discretization will result in a very sparsely connected BN due to regularization

constraints, ultimately not reflecting the interactions we are interested in. On the other hand, too rough a discretization may not provide the “user” with the desired degree of resolution required for proper decision support. Various discretization approaches have been proposed in literature, (Friedman and Goldszmidt, 1996; Monti and Cooper, 1998). Inspired by the latter we present a straightforward extension to the BN MAP learning score, allowing us to determine the “fitness” of a BN *and* discretization simultaneously, conditional on continuous data.

From now on, a superscript  $c$  denotes the continuous counterpart of a discretized variable/configuration, e.g.,  $\mathbf{d}^c$  is the original continuous data and  $\mathbf{d}$  a discretized version thereof. Let  $\Lambda$  define the discretization, that is, the set of interval boundary points for all variables; a configuration  $\lambda$  will thus “bin” the original data  $\mathbf{d}^c$  yielding  $\mathbf{d}$ . Moreover, assume that we have  $P(\mathbf{d}^c|\mathbf{d}, \Lambda)$ ; this is the generative model for the continuous data given some discretized version thereof as defined by  $\Lambda$ . Note that this model is unrelated to the BN; more on this distribution shortly. It follows that the likelihood for observing  $\mathbf{d}^c$  for a given discretization, network structure and parameters (the BN) can be written as

$$P(\mathbf{d}^c|DAG, \Theta, \Lambda) = P(\mathbf{d}|DAG, \Theta, \Lambda) P(\mathbf{d}^c|\mathbf{d}, \Lambda).$$

Embedded in a Bayesian context, we are now seeking the MAP of the posterior

$$\begin{aligned} & P(DAG, \Theta, \Lambda | \mathbf{d}^c) \\ & \propto P(\mathbf{d}^c | DAG, \Theta, \Lambda) P(DAG, \Theta, \Lambda) \\ & = \underbrace{P(\mathbf{d} | DAG, \Theta, \Lambda)}_1 \underbrace{P(\mathbf{d}^c | \mathbf{d}, \Lambda)}_2 \times \quad (2) \\ & \underbrace{P(\Theta | DAG, \Lambda)}_3 \underbrace{P(\Lambda | DAG)}_4 \underbrace{P(DAG)}_5. \end{aligned}$$

The product of the terms 1, 3 and 5 is equivalent to the (joint) BN posterior as introduced in Section 3 (terms 1 and 3 now of course depend on the discretization). Let term 4 be uniform on the space of all possible discretizations, allowing us to ignore this factor in the MAP estimation.

We define  $P(X_i^c | X_i, \Lambda_i)$  according to the following considerations: the discrete  $X_i$  is associated with several interval boundaries, such that each state  $x_i$  has a lower  $\lambda_{x_i}$  and upper boundary  $\lambda^{x_i}$ . In between this interval  $X_i^c$  is distributed uniformly, outside it is zero;  $x_i$  thus “picks” the uniform interval in which  $X_i^c$  can lie. Effectively we arrive at term 2

$$P(\mathbf{d}^c | \mathbf{d}, \Lambda) = \prod_i \prod_{x_i} \left( \frac{1}{\lambda^{x_i} - \lambda_{x_i}} \right)^{n(x_i)},$$

which leads to a preference of small intervals, while term 1 and 3 counteract the formation of a high number of intervals.

Discretization and BN learning are nested iteratively: learn a BN for a given discretization, followed by learning a new discretization, and so on. For a given discretization maximizing (2) with respect to the BN-pair ( $DAG, \Theta$ ) is equivalent to maximizing the MAP BN score alone (which is the same as learning the DAG via  $S(DAG | \mathbf{d}, \lambda)$  also implying the BN parameter estimates) because term 2 is independent of the BN. To find the interval boundaries for the discretization, the variables are discretized iteratively until (2) stops improving, or until a predefined number of iterations has been reached. At each step we select a variable and employ a binary search for the “best” discretization fixing the intervals for the other variables.

#### 4.1 A Single Continuous Target

The “optimal” discretization will not necessarily result in the required resolution for a particular target variable of interest. In our case  $rloss$  is of primary interest, and the discretization described in the last section leads to a discretization into 5 intervals. To achieve a finer resolution we treat, once we learned the BN, the target variable  $X_i$  as discrete with a very large number of states, defined by splitting the range of  $X_i^c$  into  $n_{X_i}$  intervals, e.g.,  $n_{X_i} = 512$ . For simplicity we use equidistant intervals of width  $\Delta_{X_i}$ . The states of  $X_i$  are the midpoints of the corresponding intervals. Because of the large number of states, the estimator (1) for  $\theta_{X_i | \mathbf{x}_{Pa(i)}}$ , or  $\theta_{X_j | \mathbf{x}_{Pa(j)}}$  when  $X_i \in \mathbf{X}_{Pa(j)}$ , is based on very few observations leading to weak estimates.

To avoid this problem, we use a Gaussian kernel density estimator to adopt the parameter estimation in the following manner. For  $\theta_{X_i|\mathbf{x}_{Pa(i)}}$  we set

$$\hat{\theta}_{x_i|\mathbf{x}_{Pa(i)}} = \Delta_{X_i} \tilde{P}_{X_i^c|\mathbf{x}_{Pa(i)}}(x_i),$$

where  $\tilde{P}_{X_i^c|\mathbf{x}_{Pa(i)}}(x_i) =$

$$\frac{1}{n(\mathbf{x}_{Pa(i)}) \sqrt{2\pi} h} \sum_{k|\mathbf{x}_{Pa(i)}^{(k)}=\mathbf{x}_{Pa(i)}} \exp\left(-\frac{(x_i^{c(k)} - x_i)^2}{2h^2}\right)$$

is the Gaussian kernel density estimator, with a bandwidth,  $h$ , according to Silverman's "rule of thumb" (Silverman, 1986), over all observations of  $X_i^c$ , for which  $\mathbf{X}_{Pa(i)} = \mathbf{x}_{Pa(i)}$ .

For  $X_i \in \mathbf{X}_{Pa(j)}$  we use Bayes theorem to rewrite

$$\begin{aligned} p(X_j|\mathbf{X}_{Pa(j)}) &= \frac{p(X_j, \mathbf{X}_{Pa(j)})}{\sum_{x_j} p(x_j, \mathbf{X}_{Pa(j)})} \\ &= \frac{p(X_i|X_j, \mathbf{X}_{Pa(j)-X_i}) p(X_j, \mathbf{X}_{Pa(j)-X_i})}{\sum_{x_j} p(X_i|x_j, \mathbf{X}_{Pa(j)-X_i}) p(x_j, \mathbf{X}_{Pa(j)-X_i})} \\ &= \frac{p(X_i|X_j, \mathbf{X}_{Pa(j)-X_i}) p(X_j|\mathbf{X}_{Pa(j)-X_i})}{\sum_{x_j} p(X_i|x_j, \mathbf{X}_{Pa(j)-X_i}) p(x_j|\mathbf{X}_{Pa(j)-X_i})}, \end{aligned}$$

and we set

$$\begin{aligned} \hat{\theta}_{x_j|\mathbf{x}_{Pa(j)}} &= \\ \frac{\Delta_{X_i} \tilde{P}_{X_i^c|x_j, \mathbf{x}_{Pa(j)-X_i}}(x_i) \hat{\theta}_{x_i|\mathbf{x}_{Pa(j)-X_i}}}{\sum_{x_j} \Delta_{X_i} \tilde{P}_{X_i^c|x_j, \mathbf{x}_{Pa(j)-X_i}}(x_i) \hat{\theta}_{x_i|\mathbf{x}_{Pa(j)-X_i}}}. \end{aligned}$$

Because of the large number of states for the target variable, inference can become time and space consuming. For relatively small/sparse networks this is not a big issue *per se* and in our particular case it has not posed any significant problem.

Mixtures of truncated Exponentials (MTE) are an alternative to approximate continuous distributions (Moral et al., 2001). Moreover using MTEs efficient inference is possible (Langseth et al., 2009). Methods for their construction are given by e.g. Rumí et al. (2006) and Langseth et al. (2010). However, finding

an optimal MTE-representation for a conditional/multivariate distribution from data is no trivial task. Moreover, learning both network structure and MTEs simultaneously from data is even more challenging. We leave this task for future work.

## 5 Results

We apply the BN-learning and discretization methods which are described in the last sections to the data set of Section 2. Continuous and ordinal variables are treated in the same manner and both discretized. Thus, the numbers of states is reduced for continuous as well as for discrete variables. Only the number of states for the nominal variables (*ws*, *kh*, *bt*, *own*) remains as given.

For *rloss* the majority of the observations is gathered close to the lower domain boundary; taking the logarithm of *rloss* results in more equal spread over the domain. To avoid an infinite domain range, the lower boundary, which corresponds to buildings with no damage, was set to  $\log(5.5 \cdot 10^{-6})$ , where  $5.5 \cdot 10^{-5}$  is the minimal observed loss ratio of damaged buildings.

There are missing values in the data, likely *missing (completely) at random*, M(C)AR. For convenience, we for now simply replace them by sampling from the observed values of the corresponding variable. Principled iterative methods like Expectation Maximization (EM) are intractable for our purpose, since learning both discretization and the BN means collecting sufficient statistics via inference (in the E-step of EM) disproportionately often. In the future the Markov Blanket Predictor (Riggelsen, 2006) will be employed, which is a fast one-pass approximation to EM, by restricting the attention to predictors of the Markov Blanket of a variable with a missing observation.

Figure 1 (left) shows the BN learned for all variables listed in Table 1, starting from an initial "Tree Augmented Naive Bayes" network that was obtained using the method described in (Vogel et al., 2012) with *rloss* as class variable. Some of the variables (*wte* and *fe*) have

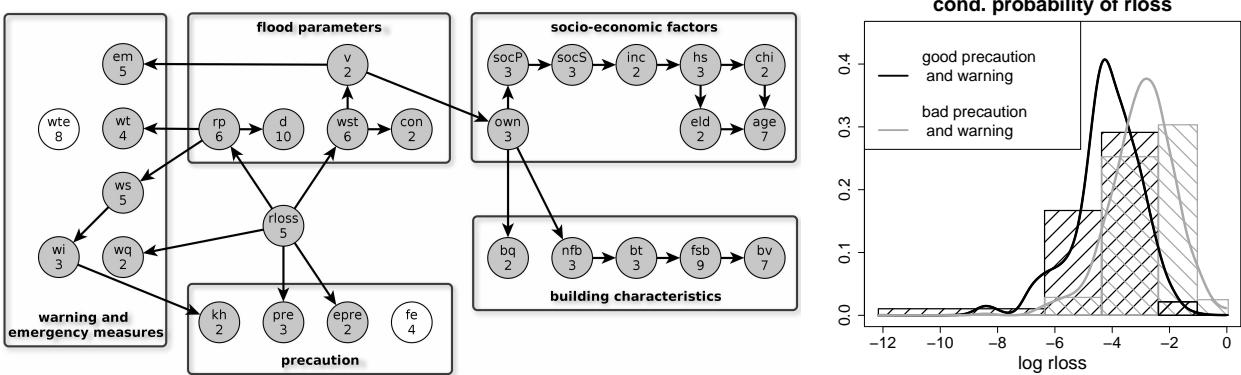


Figure 1: *left*: Bayesian network learned over the discretized data; Numbers in the nodes give the number of discrete states learned.

*right*: Conditional Probabilities of *rloss* for specific flood events using a coarse automatically learned discretization (shaded histograms) and interval refinement according to Section 4.1 (continuous lines).

zero in/out-degree as the data apparently did not support any interactions with other variables. Variables that belong to the same sub-domain, are in most cases linked via a short path. Even though all domains are included in the network, there is a clear distinction of the domains visible. Building characteristics and socio-economic factors have only indirect impact on the relative building loss, while flood parameters and precautions are closely related to the target variable. This gives us an idea about the importance of the variables for the calculation of the relative building loss.

After the network was learned, *rloss* is selected as target variable and the number of states redefined viz. Section 4.1. Thus, we get an almost continuous approximation of the conditional probability function of *rloss*. Figure 1 (right) illustrates the effect of the interval refinement. It shows the conditional distribution of *rloss* for the fine discretization in contrast to the coarse one for a flood event with water depth between 9 cm and 100 cm, a return period between 1 and 99 years and different precaution and warning levels (good:  $1 \leq wq \leq 2$ ,  $13 \leq pre \leq 38$ ,  $1 \leq epre \leq 5$ ; bad:  $3 \leq wq \leq 6$ ,  $0 \leq pre \leq 2$ ,  $epre = 6$ ).

We compare the performance of the BN in terms of the *rloss*-prediction to flood damage

assessment approaches currently used in Germany, namely to the stage-damage-function approach and to FLEMOPs+r (Elmer et al., 2010). For the stage-damage function approach, a root function is fitted to the damage data of certain object classes using least squares, i.e., the relative damage is a function of the water depth only. FLEMOPs+r has been developed using the same data set and it has been shown to provide superior results compared to other approaches currently used in Germany. FLEMOPs+r calculates the building loss ratio for private households using five classes of inundation depth, three intervals of flood frequency, three individual building types, two classes of building quality, three classes of contamination and three classes of private precaution. In essence, the data set is stratified into 27 subsamples and the average loss ratio is used as damage estimator (Elmer et al., 2010).

Additionally we compare the learned BN to the Naive Bayes (NB) and Tree Augmented Naive Bayes (TAN) learned from the same data set. These models are set up as restricted BNs with one single target variable in mind. We refer to (Vogel et al., 2012) for a corresponding description of an automatic discretization and interval refinement according to Section 4.1. Of course the independence restrictions imposed by

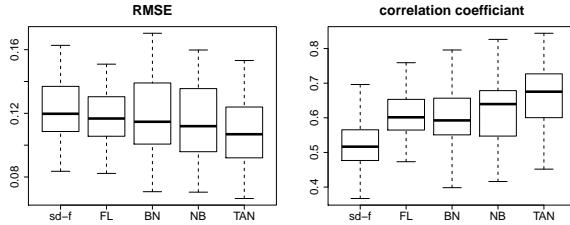


Figure 2: Comparison of flood damage estimation models (sd-f: stage damage function; FL: FLEMOps+r – model developed from same data set; BN: Bayesian Network; NB: Naive Bayes; TAN: Tree Augmented Naive Bayes).

these models do not (unlikely) obey “reality” and can not be used to gain insight into the “workings” of the underlying system. However, in terms of predictions for a single target, they have shown to often outperform BNs.<sup>1</sup>

To be able to compare to FLEMOps+r, we follow a evaluation policy commonly used in the field of hydrology: 100 bootstrap samples, each with 100 households, are drawn from the data set. We complete the results of the stage-damage-function and the FLEMOps+r model with the *rloss*-predictions we get from the BN, NB and TAN by using the expectation of the conditional *rloss*-distribution as predicted value. The predictions are quantified by the root mean squared error (RMSE) and the Pearson correlation coefficient.

It is important to stress that no separate test-sets are used: the bootstrap policy described uses parts of the training data for performance evaluation. This is not legitimate *per se*, and may influence the results considerably (optimistically). However, since the number of free parameters in the stage-damage function and the FLEMOps+r model are relatively small and the BN MAP criterion accounts for model complexity (it regularizes) these models will not over-fit and consequently performance testing on the training data will not yield overly optimistic results. Moreover, the BN MAP score is

<sup>1</sup>In fact, discriminative models are even more likely to perform well, e.g., logistic regression.

not a fitness measure of predictive performance of any target variable in particular, but rather, provides the predictive performance “overall” for all variables jointly. For the NB and TAN networks the number of free parameters is quite large and over-fitting might be a problem meaning that they on separate test-set may perform less well.

Figure 2 shows the performance measures for the 100 bootstrap samples in boxplots. It indicates that in terms of predicting *rloss* the BN performs well compared to the FLEMOps+r (the “best” method currently in use). The Naive Bayes and especially the Tree Augmented Naive Bayes show an improvement in the *rloss* prediction. However, it is important to note that the BN model in fact provides us with the joint distribution (the “correct” (in)dependence relationships) and it is therefore somewhat unfair to compare directly with approaches trying to improve upon the predictive performance of a single target variable only.

## 6 Conclusion

A BN has been learned from real-life data describing flood related observations on 29 variables, the majority continuous and some discrete. In general continuous variables pose a challenge, and often discretization is performed as a pre-processing step prior to BN model selection. We have extended the BN MAP model selection metric to score not only BNs but simultaneously take the proper discretization into account, providing an entirely data-driven approach to learn from a Bayesian *maximum a posteriori* (MAP) perspective. From a data mining point of view, the BN indeed does reveal and confirm non-trivial interactions. The learned network captures the (in)dependencies revealing connectivity between flood loss and warning, emergency measures and socio-economic factors, which are widely neglected in flood risk assessment. Additionally, from a prediction point of view where the performance of a particular target is of interest, kernel estimation improves upon the BN “multivariate” view by increasing the degree of resolution (number of

states) required for proper decision support (often derived/dependent on a single target variable). Compared to existing primarily deterministic flood damage estimation procedures, the BN shows a comparable performance with the added benefit of capturing and reasoning under uncertainty.

## Acknowledgements

This work is supported by the Potsdam Research Cluster for Georisk Analysis, Environmental Change and Sustainability, PROGRESS, a joint project of university and external organizations in the region of Potsdam-Berlin (Germany).

## References

- Lilian Blaser, Matthias Ohrnberger, Carsten Riggelsen, Andrey Babeyko, and Frank Scherbaum. 2011. Bayesian networks for tsunami early warning. *Geophysical Journal International*, 185(3):1431–1443.
- Robert Castelo and Tonas Kocka. 2003. On inclusion-driven learning of Bayesian networks. *The Journal of Machine Learning Research*, 4:527–574.
- Florian Elmer, Annegret H. Thieken, Ina Pech, and Heidi Kreibich. 2010. Influence of Flood Frequency on Residential Building Losses. *Natural Hazards and Earth System Sciences*, 10:2145–2159.
- Nir Friedman and Moises Goldszmidt. 1996. Discretizing Continuous Attributes While Learning Bayesian Networks. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 157–165.
- Nicolas M. Kuehn, Carsten Riggelsen, and Frank Scherbaum. 2011. Modeling the Joint Probability of Earthquake, Site, and Ground-Motion Parameters Using Bayesian Networks. *Bulletin of the Seismological Society of America*, 101(1):235–249.
- Helge Langseth, Thomas D. Nielsen, Rafael Rumí, and Antonio Salmerón. 2009. Inference in hybrid Bayesian networks. *Reliability Engineering & System Safety*, 94(10):1499–1509.
- Helge Langseth, Thomas D Nielsen, Rafael Rumí, and Antonio Salmerón. 2010. Parameter estimation and model selection for mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 51(5):485–498.
- Bruno Merz, Heidi Kreibich, Reimund Schwarze, and Annegret H. Thieken. 2010. Review article "Assessment of economic flood damage". *Natural Hazards and Earth System Science*, 10(8):1697–1724.
- Stefano Monti and Gregory F. Cooper. 1998. A multivariate discretization method for learning Bayesian networks from mixed data. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence UAI'98*, pages 404–413.
- Serafín Moral, Rafael Rumí, and Antonio Salmerón. 2001. Mixtures of Truncated Exponentials in Hybrid Bayesian Networks. In Salem Benferhat and Philippe Besnard, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 156–167.
- Carsten Riggelsen, Matthias Ohrnberger, and Frank Scherbaum. 2007. Dynamic bayesian networks for real-time classification of seismic signals. In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, pages 565–572.
- Carsten Riggelsen. 2006. Learning Bayesian Networks from Incomplete Data: An Efficient Method for Generating Approximate Predictive Distributions. In *SIAM International conf. on data mining*, pages 130–140.
- Carsten Riggelsen. 2008. Learning Bayesian Networks: A MAP Criterion for Joint Selection of Model Structure and Parameter. In *ICDM, 2008 Eighth IEEE International Conference on Data Mining*, pages 522–529.
- Rafael Rumí, Antonio Salmerón, and Serafín Moral. 2006. Estimating mixtures of truncated exponentials in hybrid bayesian networks. *Test*, 15(2):397–421.
- Bernard W. Silverman. 1986. *Density Estimation for Statistics and Data Analysis*, volume 26.
- Annegret H. Thieken, Meike Müller, Heidi Kreibich, and Bruno Merz. 2005. Flood damage and influencing factors: New insights from the August 2002 flood in Germany. *Water resources research*, 41.
- Kristin Vogel, Carsten Riggelsen, Nicolas Kuehn, and Frank Scherbaum. 2012. Graphical Models as Surrogates for Complex Ground Motion Models. *Proceedings of the 11th International Conference on Artificial Intelligence and Soft Computing*.

# Computationally efficient probabilistic inference with noisy threshold models based on a CP tensor decomposition\*

Jiří Vomlel

Institute of Information Theory and Automation of the AS CR  
Prague, Czech Republic  
vomlel@utia.cas.cz

Petr Tichavský

Institute of Information Theory and Automation of the AS CR  
Prague, Czech Republic  
tichavsk@utia.cas.cz

## Abstract

Conditional probability tables (CPTs) of threshold functions represent a generalization of two popular models – noisy-or and noisy-and. They constitute an alternative to these two models in case they are too rough. When using the standard inference techniques the inference complexity is exponential with respect to the number of parents of a variable. In case the CPTs take a special form (in this paper it is the noisy-threshold model) more efficient inference techniques could be employed. Each CPT defined for variables with finite number of states can be viewed as a tensor (a multilinear array). Tensors can be decomposed as linear combinations of rank-one tensors, where a rank one tensor is an outer product of vectors. Such decomposition is referred to as Canonical Polyadic (CP) or CANDECOMP-PARAFAC (CP) decomposition. The tensor decomposition offers a compact representation of CPTs which can be efficiently utilized in probabilistic inference. In this paper we propose a CP decomposition of tensors corresponding to CPTs of threshold functions and their noisy counterparts. We performed experiments on subnetworks of the well-known QMR-DT network generalized by replacing noisy-or by noisy-threshold models. Each generated subnetwork contained more than one hundred variables. The results of our experiments reveal that by using the suggested decomposition of CPTs we can get computational savings in several orders of magnitude.

## 1 Introduction

In many applications of Bayesian networks (Jensen and Nielsen, 2007), conditional probability tables (CPTs) have a certain local structure. Canonical models (Díez and Druzdzel, 2006) form a commonly used class of CPTs with the local structure being defined as a combination of a deterministic part with independent probabilistic influence of each parent variable, see Figure 1.

The joint probability distribution of the Bayesian network in Figure 1 is

$$P(Y|X'_1, \dots, X'_k) \prod_{i=1}^k P(X'_i|X_i)P(X_i) ,$$

where the first term  $P(Y|X'_1, \dots, X'_k)$  corresponds to a deterministic function and terms  $P(X'_i|X_i)$  to the probabilistic part (often called noise). We can replace the Bayesian network of Figure 1 by a model without auxiliary variables  $X'_1, \dots, X'_k$  by marginalizing them out from the Bayesian network. The values of  $P(Y|X_1, \dots, X_k)$  can be computed from the

\*This work was supported by Czech Science Foundation through projects 201/08/0539 and 102/09/1278.

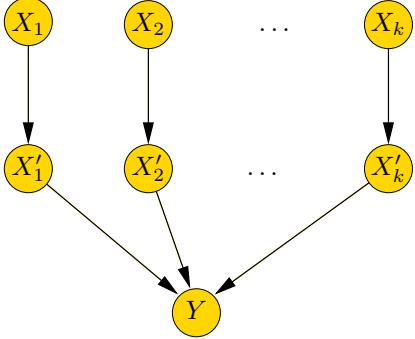


Figure 1: A Bayesian network with a canonical model with explicit deterministic part  $P(Y|X'_1, \dots, X'_k)$  and probabilistic parts  $P(X'_i|X_i), i = 1, \dots, k$ .

original model by

$$P(Y|X_1, \dots, X_k) = \sum_{X'_1} \dots \sum_{X'_k} P(Y, X'_1, \dots, X'_k) \cdot \prod_{i=1}^k P(X'_i|X_i).$$

Assume CPT with the state  $y$  of variable  $Y$  being observed. As it was suggested in (Savicky and Vomlel, 2007) we can rewrite each CPT as a product of two-dimensional potentials  $\psi_i, i = 1, \dots, k$

$$P(y|X_1, \dots, X_k) = \sum_B \prod_{i=1}^k \psi(B, X_i), \quad (1)$$

where  $B$  is an auxiliary variable. This transformation can be visualized by the undirected graph given in Figure 2.

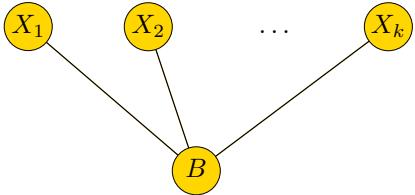


Figure 2: Model of  $P(y|X_1, \dots, X_k)$  after the transformation using auxiliary variable  $B$ .

In order to guarantee the above equality, variable  $B$  has to have certain number of states.

However, the equality can be always satisfied if the number of states of  $B$  is the product of the number of states of variables  $X_1, \dots, X_k$ . The transformation becomes computationally advantageous if the number of states is low, which is the case of CPTs of canonical models. It was observed in (Savicky and Vomlel, 2007) that since each CPT can be understood as a tensor<sup>1</sup> the minimum number of states of  $B$  equals the rank of tensor  $A$  whose values are defined as

$$A_{i_1, \dots, i_k} = P(y|X_1 = x_{i_1}, \dots, X_k = x_{i_k}),$$

for all combinations of states  $(x_{i_1}, \dots, x_{i_k})$  of variables  $X_1, \dots, X_k$ . The decomposition of tensors into the form corresponding to the right hand side of formula (1) has been studied for more than forty years (Carroll and Chang, 1970; Harshman, 1970) and it is known now as Canonical Polyadic (CP) or CANDECOMP-PARAFAC (CP) decomposition. In (Comon et al., 2008) it is called an outer-product decomposition.

In this paper we deal with conditional probability tables representing one specific type of canonical models – deterministic threshold functions and their noisy counterparts. An  $(\ell, k)$  threshold function is a function of  $k$  binary arguments that takes the value one if at least  $\ell$  out of its  $k$  arguments take value one – otherwise the function value is zero. The noisy version allows noise at the inputs of the function. The noisy threshold models represent a generalization of two popular models - noisy-or and noisy-and. They constitute an alternative to noisy-or and noisy-and in case they are too rough. The conditional probability tables of the threshold functions appear, for example, in medical applications of Bayesian networks (Visscher et al., 2009; van Gerven et al., 2007; Jurgeleit et al., 2006; Jurgeleit and Heskes, 2006).

For CP tensor decompositions of other canonical models see Vomlel (2011) where the tensors of  $\ell$ -out-of- $k$  functions are studied and Savicky and Vomlel (2007), where CP decompositions of

<sup>1</sup>The formal definition of a tensor can be found in the next section.

tensors of several other canonical models (noisy-max, noisy-min, noisy-add, noisy-xor) are described.

The rest of this paper is organized as follows. In Section 2 we introduce the necessary tensor notation, define tensors of the threshold functions, and present their basic properties. Section 3 represents the main original contribution of this paper. We propose an algorithm for the CP decomposition of tensors of the threshold functions based on the upper bound of the symmetric rank of these tensors. In Section 4 we conclude the paper by computational comparisons performed on a generalized version of the QMR-DT network.

## 2 Preliminaries

Tensor is a mapping<sup>2</sup>  $\mathbf{A} : \mathbb{I} \rightarrow \mathbb{R}$ , where  $\mathbb{I} = I_1 \times \dots \times I_k$ ,  $k$  is a natural number called the order of tensor  $\mathbf{A}$ , and  $I_j, j = 1, \dots, k$  are index sets. Typically,  $I_j$  are sets of integers of cardinality  $n_j$ . Then we can say that tensor  $\mathbf{A}$  has dimensions  $n_1, \dots, n_k$ . In this paper all index sets will be  $I_j = \{0, 1\}, j = 1, \dots, k$ .

Tensor  $\mathbf{A}$  has rank one if it can be written as an outer product of vectors, i.e.,

$$\mathbf{A} = \mathbf{a}_1 \otimes \dots \otimes \mathbf{a}_k ,$$

where  $\mathbf{a}_j, j = 1, \dots, k$  are real valued vectors of length  $|I_j|$ .

Each tensor can be decomposed as a linear combination of rank-one tensors:

$$\mathbf{A} = \sum_{i=1}^r b_i \cdot \mathbf{a}_{i,1} \otimes \dots \otimes \mathbf{a}_{i,k} , \quad (2)$$

The rank of a tensor  $\mathbf{A}$ , denoted  $rank(\mathbf{A})$ , is the minimal  $r$  over all such decompositions. The decomposition of a tensor  $\mathbf{A}$  to tensors of rank one that sum up to  $\mathbf{A}$  is called CP tensor decomposition.

A special class of tensors that appear in the problems that motivated our research in this area (Savicky and Vomlel, 2007; Vomlel, 2002)

<sup>2</sup>Often tensor values are from  $\mathbb{C}$ . However in this paper we will restrict them to be from  $\mathbb{R}$ .

are tensors representing functions. In this paper we will pay special attention to tensors representing the threshold function, i.e. a Boolean function taking value 1 if and only if  $\ell$  of more of its  $k$  inputs have value 1.

**Definition 1.** Tensor  $\mathbf{T}(\ell, k) : \{0, 1\}^k \rightarrow \{0, 1\}$  represents an  $(\ell, k)$ -threshold function if it holds for  $(i_1, \dots, i_k) \in \{0, 1\}^k$ :

$$\begin{aligned} \mathbf{T}_{i_1, \dots, i_k}(\ell, k) &= \delta(i_1 + \dots + i_k \geq \ell) \\ \delta(i \geq \ell) &= \begin{cases} 1 & \text{if } i \geq \ell \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

**Example 1.**

$$\mathbf{T}(2, 4) = \begin{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \end{pmatrix} .$$

Tensors representing the threshold function have certain nice properties, e.g., they are symmetric.

**Definition 2.** Tensor  $\mathbf{A} : \{0, 1\}^k \rightarrow \mathbb{R}$  is symmetric if for  $(i_1, \dots, i_k) \in \{0, 1\}^k$  it holds that

$$\mathbf{A}_{i_1, \dots, i_k} = \mathbf{A}_{i_{\sigma(1)}, \dots, i_{\sigma(k)}} ,$$

for any permutation  $\sigma$  of  $\{1, \dots, k\}$ .

For symmetric tensors it is possible to define a symmetric rank as follows.

**Definition 3.** The symmetric rank  $srank(\mathbf{A})$  of a tensor  $\mathbf{A}$  is the minimum number of symmetric rank-one tensors such that their linear combination equals  $\mathbf{A}$

$$\begin{aligned} \mathbf{A} &= \sum_{i=1}^r b_i \cdot \underbrace{\mathbf{a}_i \otimes \dots \otimes \mathbf{a}_i}_{k \text{ copies}} \\ &= \sum_{i=1}^r b_i \cdot \mathbf{a}_i^{\otimes k} , \end{aligned} \quad (3)$$

where we adopt the notation of (Comon et al., 2008).

*Remark.* It is not known whether it holds for symmetric tensors  $\mathbf{A}$  that  $rank(\mathbf{A}) = srank(\mathbf{A})$ .

Each symmetric tensor  $\mathbf{A} : \{0,1\}^k \rightarrow \mathbb{R}$  of rank one can be written as

$$\mathbf{A} = \begin{cases} (0, a)^{\otimes k} & \text{if } \mathbf{A}_{0,\dots,0} = 0 \\ b \cdot (1, a)^{\otimes k} & \text{otherwise,} \end{cases} \quad (4)$$

where  $a, b \in \mathbb{R}$ .

In the following lemma we treat the border cases with symmetric rank one.

**Lemma 1.** The symmetric rank of tensors  $\mathbf{T}(\ell, k)$  representing the respective  $(\ell, k)$ -threshold function for  $\ell \in \{0, k\}$  is one.

*Proof.*

$$\begin{aligned} \mathbf{T}(k, k) &= (0, 1)^{\otimes k} \\ \mathbf{T}(0, k) &= (1, 1)^{\otimes k}. \end{aligned} \quad \square$$

In the next lemma another we present a case with a low symmetric rank equal to two.

**Lemma 2.** The symmetric rank of tensors  $\mathbf{T}(1, k)$  representing the respective  $(1, k)$ -threshold function is two.

*Proof.*

$$\mathbf{T}(1, k) = (1, 1)^{\otimes k} - (1, 0)^{\otimes k}$$

and there does not exist any vector  $\mathbf{a}$  such that

$$\mathbf{T}(1, k) = \mathbf{a}^{\otimes k}.$$

To see this note that  $\mathbf{T}(1, k)_{0,\dots,0} = 0$ . This requires  $\mathbf{a} = (0, a), a \in \mathbb{R}$ . But tensor  $(0, a)^{\otimes k}$  has all its values but the one at  $(1, \dots, 1)$  equal to zero and thus cannot be equal to  $\mathbf{T}(1, k)$ .  $\square$

### 3 A CP tensor decomposition

In this section we will construct a CP decomposition of the tensor  $\mathbf{T}(\ell, k)$  representing an  $(\ell, k)$ -threshold function. In the previous section we have already treated the border cases for  $\ell \in \{0, 1, k\}$ .

If we restrict the rank-one tensors in the decomposition as defined by formula (3) to tensors with a non-zero value at the  $(0, \dots, 0)$  position then we can rewrite formula (3) as

$$\mathbf{A} = \sum_{i=1}^r b_i \cdot (1, a_i)^{\otimes k}, \quad (5)$$

where  $a_i, b_i \in \mathbb{R}, i = 1, \dots, k$ . Note that, generally, this restricted decomposition need not be

a minimal decomposition of  $\mathbf{A}$  (with respect to  $r$ ). On the other hand, it can be used to provide an upper bound on the symmetric rank of a tensor  $\mathbf{A}$ .

It follows from formula (5) that a sufficient condition for a symmetric decomposition of a tensor of an  $(\ell, k)$ -threshold function to have rank  $r$  is the following system of equations:

$$\begin{aligned} a_1^0 \cdot b_1 + \dots + a_r^0 \cdot b_r &= \delta(0 \geq \ell) \\ a_1^1 \cdot b_1 + \dots + a_r^1 \cdot b_r &= \delta(1 \geq \ell) \\ &\vdots \\ a_1^k \cdot b_1 + \dots + a_r^k \cdot b_r &= \delta(k \geq \ell), \end{aligned} \quad (6)$$

which is a system of  $k+1$  equations with  $2r$  variables.

Let  $m \in \mathbb{N}^+$ ,  $\mathbf{a} = (a_1, \dots, a_m), a_j \in \mathbb{R}, j \in \{1, \dots, m\}$ , and  $V(\mathbf{a})$  be a Vandermonde matrix  $m \times m$  defined as

$$V(\mathbf{a}) = \begin{pmatrix} 1 & \dots & 1 \\ a_1 & \dots & a_m \\ a_1^{m-1} & \dots & a_m^{m-1} \end{pmatrix}.$$

Further let  $\mathbf{e}(\ell)$  be the vector of length  $k$  having its values  $e_i(\ell)$  at positions  $i = 1, \dots, \ell$  equal to zero and for  $i = \ell+1, \dots, k$  equal to one i.e.,

$$\mathbf{e}(\ell) = (\delta(0 \geq \ell), \dots, \delta(k-1 \geq \ell))^T. \quad (7)$$

A sufficient condition for the solution of the system (6) for  $r = k$  is to solve the following system of  $3k$  equations. Let  $\mathbf{a} = (a_1, \dots, a_k)$ ,  $\mathbf{b} = (b_1, \dots, b_k)$ ,  $\mathbf{c} = (c_1, \dots, c_k)$ , and  $\mathbf{a} * \mathbf{b}$  denote elementwise multiplication (Hadamard product) of vectors  $\mathbf{a}$  and  $\mathbf{b}$ . All but the last equation of system (6) correspond to system (8), all but the first equation of system (6) correspond to systems (9) and (10):

$$V(\mathbf{a}) \cdot \mathbf{b} = \mathbf{e}(\ell) \quad (8)$$

$$V(\mathbf{a}) \cdot \mathbf{c} = \mathbf{e}(\ell-1) \quad (9)$$

$$\mathbf{c} = \mathbf{a} * \mathbf{b}. \quad (10)$$

If values of  $a_i, i = 1, \dots, k$  are distinct then

$$\mathbf{b} = V(\mathbf{a})^{-1} \cdot \mathbf{e}(\ell) \quad (11)$$

$$\mathbf{c} = V(\mathbf{a})^{-1} \cdot \mathbf{e}(\ell-1). \quad (12)$$

Note that the explicit formula for the inverse of the Vandermonde matrix is known. Let

$$p(x) = \prod_{j=1}^k (x - a_j) \text{ and} \quad (13)$$

$$p_i(x) = \prod_{j \neq i} (x - a_j) \quad (14)$$

be polynomials in variable  $x$ . Further let  $p[j], j = 1, \dots, k+1$  denote the coefficient in the term with  $x^{j-1}$  of polynomial  $p(x)$  so that

$$p(x) = \sum_{j=1}^{k+1} p[j] \cdot x^{j-1}.$$

Finally, let  $p|_{x=a_i}$  denote the substitution of  $a_i$  in place of  $x$  in polynomial  $p(x)$ . We can write equations (11) and (12) for  $i = 1, \dots, k$  as

$$b_i = \frac{\sum_{j=\ell+1}^k p_i[j]}{p_i|_{x=a_i}} \quad (15)$$

$$c_i = \frac{\sum_{j=\ell}^k p_i[j]}{p_i|_{x=a_i}}. \quad (16)$$

Substituting (15) and (16) into (10) we get for  $i = 1, \dots, k$ :

$$a_i = \frac{c_i}{b_i} = \frac{\sum_{j=\ell}^k p_i[j]}{\sum_{j=\ell+1}^k p_i[j]}, \quad (17)$$

where the right hand side depends on  $a_j, j \neq i$  only. Due to symmetry, if one equation of (17) holds then all equations hold and the system can be reduced to one equation only, e.g., to

$$a_k = \frac{\sum_{j=\ell}^k p_k[j]}{\sum_{j=\ell+1}^k p_k[j]}, \quad (18)$$

where the right hand side depends on  $a_1, \dots, a_{k-1}$ . If we set  $a_1, \dots, a_k$  so that they are pairwise distinct,  $a_k$  satisfies (18), and from (11) we compute  $b_1, \dots, b_k$ , which is under the above constraints always possible, then we have a solution of system (6). As a consequence we have following lemma.

**Lemma 3.** *The symmetric rank of a tensor  $\mathbf{T}(\ell, k)$  representing  $(\ell, k)$ -threshold function for  $\ell = 2, \dots, k-1$  is at most  $k$ .*

Table 1: An algorithm for the CP tensor decomposition of tensors  $\mathbf{T}(\ell, k)$  defined by formula (5) for  $r = k$ .

---

|         |                                                                            |
|---------|----------------------------------------------------------------------------|
| Input:  | $k, \ell$                                                                  |
| Output: | $(a_1, \dots, a_k)$ and $(b_1, \dots, b_k)$                                |
| Find    | $\mathbf{a}_0 = (a_1, \dots, a_k)$ at random such that:                    |
|         | $a_i \neq a_j, i \neq j$                                                   |
|         | $\sum_{j=\ell+1}^k p_k[j] \neq 0$                                          |
|         | $a_k = \frac{\sum_{j=\ell}^k p_k[j]}{\sum_{j=\ell+1}^k p_k[j]}$            |
| Find    | $\mathbf{a} = (a_1, \dots, a_k)$ minimizing $\kappa(V(\mathbf{a}))$        |
|         | subject to $a_k = \frac{\sum_{j=\ell}^k p_k[j]}{\sum_{j=\ell+1}^k p_k[j]}$ |
|         | starting at initial point $\mathbf{a}_0$                                   |
| For     | $i \in \{1, \dots, k\}$ compute:                                           |
|         | $b_i \leftarrow \frac{\sum_{j=\ell+1}^k p_i[j]}{p_i _{x=a_i}}$             |

---

*Remark.* Lemma 3 slightly lowers the general upper bound for symmetric tensors from (Comon et al., 2008, Section 4.1) for tensors of  $(\ell, k)$ -threshold function. With complex numbers being allowed in the decomposition their upper bound for symmetric tensors with all dimensions being two is  $k+1$ .

The main contribution of this paper is the construction of a CP tensor decomposition of any tensor  $\mathbf{T}(\ell, k)$  representing the respective  $(\ell, k)$ -threshold function for  $\ell = 2, \dots, k-1$  to the sum of  $k$  symmetric tensors of rank-one. In Table 1 we summarize the algorithm for the CP decomposition defined by formula (5) for  $r = k$ .

Initial values of  $a_1, \dots, a_{k-1}$  are drawn at random from Gaussian distribution with mean  $\mu = 0$  and variance  $\sigma^2 = 1$ . If  $(a_1, \dots, a_{k-1})$  are such that either  $(a_1, \dots, a_k)$  are not pairwise distinct or  $\sum_{j=\ell+1}^k p_k[j] = 0$  then a new configuration is generated. In our experiments, we never had to generate new values of  $(a_1, \dots, a_{k-1})$ . But since there is no guarantee that it cannot happen therefore the loop is needed. From the computational point of view it is advantageous to have  $\mathbf{a}$  defined so that the condition number  $\kappa(V(\mathbf{a}))$  of matrix  $V(\mathbf{a})$  is minimized.

Since the values of  $\mathbf{a}$  and  $\mathbf{b}$  can be precomputed one can spend some time with an optimization algorithm searching for values  $\mathbf{a}$  minimizing  $\kappa(V(\mathbf{a}))$ . We experimented with Nelder-Mead method restarted from different starting points. Note that if  $a_1, \dots, a_k$  are distinct then for  $i = 1, \dots, k$   $p_i|_{x=a_i} \neq 0$  and  $b_i$  are well-defined. Around  $k = 25$  Vandermonde matrices get easily badly conditioned. However, in the numerical experiments we have not observed any significant errors<sup>3</sup> even in computations performed with CPTs for  $(\ell, k)$ -threshold function with  $k = 25, 26, 27^4$ .

In the next example we will show that, generally, Lemma 3 does not provide a tight upper bound even for  $\ell \in 2, \dots, k - 1$ .

### Example 2.

$$\begin{aligned} \mathbf{T}(2, 5) &= (1, -\frac{1}{2})^{\otimes 5} - 3 \cdot (1, \frac{1}{2})^{\otimes 5} \\ &\quad + (1 - \frac{\sqrt{3}}{2}) \cdot (1, -\frac{\sqrt{3}}{2})^{\otimes 5} \\ &\quad + (1 + \frac{\sqrt{3}}{2}) \cdot (1, \frac{\sqrt{3}}{2})^{\otimes 5} \end{aligned}$$

which implies that for  $k = 5$

$$\text{srank}(\mathbf{T}(2, 5)) \leq 4 = k - 1 < k .$$

On the other hand there exist tensors for which Lemma 3 provides a tight upper bound.

**Lemma 4.** *The symmetric rank of tensors  $\mathbf{T}(k-1, k)$  representing  $(k-1, k)$ -threshold function is  $k$ .*

*Proof.* We will prove that  $\text{srank}(\mathbf{T}(k-1, k)) > k - 1$  by contradiction. Assume  $r = k - 1$ . In this case the system (6) corresponds to

$$\begin{aligned} a_1^0 \cdot b_1 + \dots + a_{k-1}^0 \cdot b_{k-1} &= 0 \\ &\vdots \\ a_1^{k-2} \cdot b_1 + \dots + a_{k-1}^{k-2} \cdot b_{k-1} &= 0 \\ a_1^{k-1} \cdot b_1 + \dots + a_{k-1}^{k-1} \cdot b_{k-1} &= 1 \\ a_1^k \cdot b_1 + \dots + a_{k-1}^k \cdot b_{k-1} &= 1 . \quad (19) \end{aligned}$$

<sup>3</sup>We compared one dimensional marginal probabilities computed from the full CPTs with marginal probabilities computed in models after CP decomposition.

<sup>4</sup>Note that computational complexity with the standard method is exponential with respect to  $k$ . Therefore this method has difficulties with getting probabilities for higher  $k$ .

Let  $\mathbf{a} = (a_1, \dots, a_{k-1})$ . We can write the first  $k - 1$  equalities of (19) using a Vandermonde matrix as

$$\mathbf{V}(\mathbf{a}) \cdot \mathbf{b} = \mathbf{0} ,$$

which cannot hold unless the Vandermonde matrix is singular, which is not the case. Note that even if rank-one tensor of the form  $(0, a)^{\otimes k}$  (which we excluded from formula (5)) were allowed to be part of the decomposition then it would not add any value to the left hand side of the first  $k$  equations of system (19) since all its values except  $A_{1,\dots,1}$  are zero. Therefore  $\text{srank}(\mathbf{T}(k-1, k)) > k - 1$ . This together with Lemma 3 implies  $\text{srank}(\mathbf{T}(k-1, k)) = k$ .  $\square$

Due to Theorem 6 from (Savicky and Vomlel, 2007) the results derived for deterministic part of canonical models can be easily combined with the probabilistic part representing the noise. See Section 3.5 in (Savicky and Vomlel, 2007) for details. A consequence is that the upper bound of the symmetric rank (Lemma 3) for tensors  $\mathbf{T}(\ell, k)$  representing  $(\ell, k)$ -threshold functions is also an upper bound on the rank of their noisy counterparts.

In this section all results were derived for  $P(y|X_1, \dots, X_k)$  with  $y = 1$ . Corresponding results for  $y = 0$  can be achieved after flipping the values on each coordinate in tensors and in vectors that generate the tensors of a CP decomposition.

## 4 Experiments

We performed experiments with the Quick Medical Reference - Decision Theoretic version (QMR-DT) derived from the original QMR (Miller et al., 1986) by (Shwe et al., 1991). The Bayesian network of QMR-DT contains 570 diseases (variables  $X_i$ ) and 4075 observations (variables  $Y_j$ ). The conditional probability tables for observations given related diseases are noisy-or models. We generalized the QMR-DT by replacing noisy-or with noisy-threshold models. The experiments were performed with subnetworks of QMR-DT. In the first test, we randomly selected 14 observations. We included all their parents in the generated subnetwork.

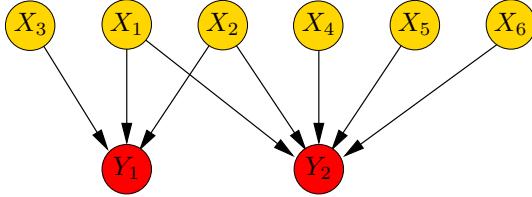


Figure 3: A part of the QMR-DT network.

In Figure 3 we give an example of a subnet-work of the QMR-DT network generated by two observations and their parents. In this way we generated forty different networks.

For each network we compared computational complexity of the junction tree method (Jensen et al., 1990) applied to models after two different transformations:

- moralization and triangulation (the standard method)
- the tensor CP decomposition applied to CPTs with number of parents higher than four<sup>5</sup> followed by triangulation.

We measured the computational complexity by the total table size of models computed by Hugin optimal triangulation<sup>6</sup>.

In the second test, we repeated the same process with 28 observations instead of 14. In both tests we have got together eighty bipar-tite graphs with their size in the range from 46 to 585 nodes. The results of experiments are summarized in Figure 4. Note the logarithmic scales. If the total table size is larger than  $2^{64}$  the models are intractable in Hugin. Numerical experiments reveal that we can get a gain in the order of several magnitudes and many intractable models became tractable.

A different approach exploiting a local structure in CPTs are arithmetic circuits (ACs) (Darwiche, 2003). In (Vomlel and Savicky, 2008) the CP tensor decomposition was used to preprocess Bayesian networks containing noisy-or models. The ACs of the prepro-

<sup>5</sup>For CPTs with less than four parents we used mor-alization instead.

<sup>6</sup>Hugin Expert A/S, <http://www.hugin.com>

cessed networks were compared with ACs created by Ace<sup>7</sup> from networks after parent divorcing. The CP tensor decomposition decreased the size of ACs for a majority of tested net-works (about 88%). We conjecture we would get similar results for experiments reported in this section. However, we did not perform these experiments – they should be a topic of a our future research along with comparisons with other methods exploiting local structure of CPTs.

## 5 Conclusions

We proposed a CP decomposition of tensors cor-responding to threshold functions. We applied this decomposition to probabilistic inference in Bayesian networks containing conditional prob-ability tables representing noisy threshold func-tions. We performed computational exper-iments with a generalized version of QMR-DT where the noisy-or models were replaced by noisy threshold models. The CP tensor decom-position led to a computational gain in the or-der of several magnitudes and made many in-tractable models manageable.

## Acknowledgments

We would like to thank Frank Jensen for the Hugin optimal triangulation method and Gre-gory F. Cooper from University of Pittsburgh for the structural part of QMR-DT model.

## References

- J. D. Carroll and J. J. Chang. 1970. Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young de-composition. *Psychometrika*, 35:283–319.
- P. Comon, G. Golub, Lek-Heng Lim, and B. Mourain. 2008. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1254–1279.
- A. Darwiche. 2003. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50:280–305.
- F. J. Díez and M. J. Druzdzel. 2006. Canonical probabilistic models for knowledge engineering.

<sup>7</sup>Ace, A Bayesian Network Compiler, 2008, <http://reasoning.cs.ucla.edu/ace/>

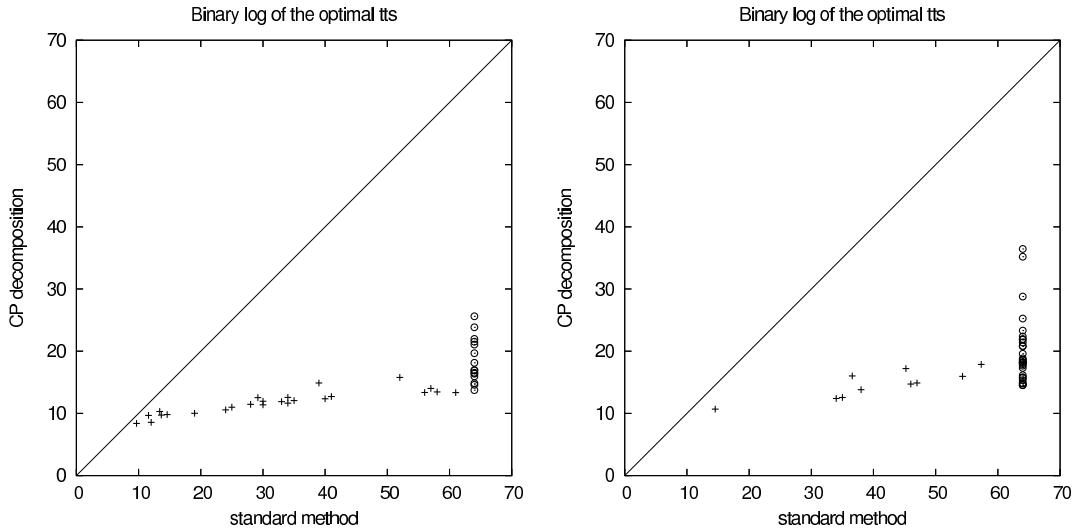


Figure 4: Comparison of total table size for QMR-DT subnetworks for the standard method and after the CP tensor decomposition. The left hand side graph is for networks after 14 observations, the right side graphs after 28 observations. The circle points positioned at the value of  $tts = 2^{64}$  of the standard method correspond to networks where the standard method failed since  $tts > 2^{64}$ .

- Technical Report CISIAD-06-01, UNED, Madrid, Spain.
- R. A. Harshman. 1970. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-mode factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84.
- F. V. Jensen and T. D. Nielsen. 2007. *Bayesian Networks and Decision Graphs*, 2nd ed. Springer.
- F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. 1990. Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly*, 4:269–282.
- R. Jurgelenaite and T. Heskes. 2006. EM algorithm for symmetric causal independence models. In *ECML'06*, volume 4212 of *Lecture Notes in Computer Science*, pages 234–245. Springer.
- R. Jurgelenaite, P. Lucas, and T. Heskes. 2006. Exploring the noisy threshold function in designing Bayesian networks. In M. Brämer, F. Coenen, and T. Allen, editors, *Research and Development in Intelligent Systems XXII*, pages 133–146. Springer.
- R. A. Miller, F. E. Fasarie, and J. D. Myers. 1986. Quick medical reference (QMR) for diagnostic assistance. *Medical Computing*, 3:34–48.
- P. Savicky and J. Vomlel. 2007. Exploiting tensor rank-one decomposition in probabilistic inference. *Kybernetika*, 43(5):747–764.
- M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper. 1991. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–255.
- M. A. J. van Gerven, R. Jurgelenaite, B. G. Taal, T. Heskes, and P. J. F. Lucas. 2007. Predicting carcinoid heart disease with the noisy-threshold classifier. *Artificial Intelligence in Medicine*, 40(1):45–55.
- S. Visscher, P. J. F. Lucas, C. A. M. Schurink, and M. J. M. Bonten. 2009. Modelling treatment effects in a clinical bayesian network using boolean threshold functions. *Artificial Intelligence in Medicine*, 46(3):251–266.
- J. Vomlel and P. Savicky. 2008. Arithmetic circuits of the noisy-or models. In Manfred Jaeger and Thomas D. Nielsen, editors, *Proceedings of PGM'08*, pages 297–304.
- J. Vomlel. 2002. Exploiting functional dependence in Bayesian network inference. In *Proceedings of UAI'02*, pages 528–535. Morgan Kaufmann Publishers.
- J. Vomlel. 2011. Rank of tensors of  $\ell$ -out-of- $k$  functions: An application in probabilistic inference. *Kybernetika*, 47(3):317–336.

# Bayesian Network Inference With NIN-AND Tree Models

Yang Xiang, University of Guelph, Canada

## Abstract

Non-impeding noisy-AND (NIN-AND) tree models were developed to improve efficiency and expressiveness in acquisition of conditional probability tables (CPTs) when constructing Bayesian networks (BNs). To take advantage of these models in the BN inference, we propose a multiplicative factorization of these models and a compilation of NIN-AND tree modeled BNs for lazy propagation (LP). Soundness of the method and its efficiency improvement are shown.

## 1 Introduction

BNs allow uncertain knowledge to be expressed with a linear number of CPTs. Each CPT, often about an effect conditioned on its  $n$  causes, is exponential in size in terms of  $n$ . Noisy-OR (Pearl, 1988) and several causal independence models (CIMs), e.g., (Heckerman and Breese, 1996; Galan and Diez, 2000; Lemmer and Gossink, 2004), reduce the number of parameters to linear, but are limited to reinforcing interactions (Xiang and Jia, 2007). This is overcome in NIN-AND tree models that allow undermining and recursive mixture of both.

Besides making acquisition easier, CIMs are explored for the inference efficiency by different approaches. These include parent divorcing (Olesen et al., 1989), temporal belief nets (Heckerman, 1993), heterogeneous factorization (Zhang and Poole, 1996), multiplicative factorization (MF) (Takikawa and D'Ambrosio, 1999), tensor rank-one decomposition (Savicky and Vomlel, 2007), among others. Most explore reinforcing CIM noisy-OR and noisy-MAX.

This work takes the approach by (Madsen and D'Ambrosio, 2000), who apply a MF to noisy-MAX for LP in junction trees (JTs). We propose a MF of binary NIN-AND tree models and a compilation of NIN-AND tree modeled BNs into JT for LP. This approach has the following properties: (a) It is based on NIN-AND tree CIMs, and expresses mixture of reinforcing and undermining interactions. (b) It is based on MFs, and does not impose constraints to

variable elimination ordering (as, e.g., heterogeneous factorization does (Zhang and Poole, 1996)). (c) It is based on LP in JT, uses the smallest runtime factors (unlike standard belief propagation in JT, e.g., (Jensen et al., 1990), where each runtime factor is a product of several CPTs), and provides exact posterior marginals for all variables. (d) Moralization, as commonly performed, is not needed, which is similar to rank-one decomposition (Savicky and Vomlel, 2007).

Sec. 2 reviews NIN-AND tree models. MFs of single NIN-AND gate models and their soundness are presented in Secs. 3 and 4. The MF of multi-gate NIN-AND tree models is covered in Sec. 5, with soundness proven in Sec. 6. How to compile factorized models for LP is described in Sec. 7. Sec. 8 demonstrates the efficiency gain.

## 2 NIN-AND Tree Causal Models

This section introduces briefly binary NIN-AND tree models, which this work focuses on. More details on these models can be found in (Xiang and Jia, 2007). An uncertain cause can produce an effect but does not always do so. Denote the effect by  $e$  with domain  $D_e = \{e^0, e^1\}$ , where  $e^1$  denotes  $e = \text{true}$ , and the set of all causes (including a leaky variable if any) of  $e$  by  $C = \{c_1, \dots, c_n\}$ , where each cause has domain  $D_{c_i} = \{c_i^0, c_i^1\}$ . At times, we write these domains exchangeably as  $D_e = D_{c_i} = \{0, 1\}$ .

A *single-causal success* is an event where  $c_i$  causes  $e$  to occur when other causes are false.

Denote the event by  $e^1 \leftarrow c_i^1$  and its probability by  $P(e^1 \leftarrow c_i^1)$ . Smoking causing lung cancer is denoted  $lc^1 \leftarrow smk^1$ . A *single-causal failure*, where  $e$  is false when  $c_i$  is true and other causes are false, is denoted by  $e^0 \leftarrow c_i^1$ . A *multi-causal success* is an event where a set  $X = \{c_1, \dots, c_k\}$  of causes ( $k \leq n$ ) cause  $e$ , and is denoted by  $e^1 \leftarrow c_1^1, \dots, c_k^1$  or  $e^1 \leftarrow \underline{x}^1$ .

A CPT  $P(e|C)$  relates to causal probabilities as follows: If  $C = \{c_1, c_2, c_3\}$ , then  $P(e^1|c_1^1, c_2^0, c_3^1) = P(e^1 \leftarrow c_1^1, c_3^1)$ .

Causal probabilities satisfy the following:

$$P(e^1 \leftarrow \emptyset) = 0 \quad (1)$$

$$P(e^0 \leftarrow \underline{x}^1) = 1 - P(e^1 \leftarrow \underline{x}^1) \quad (2)$$

Eqn. (1) denotes that the effect never occurs if all causes are false. Eqn. (2) relates causal success to failure.

Causes reinforce each other if collectively they are at least as effective as when some are active. Radiotherapy and chemotherapy are reinforcing causes for curing cancer. If collectively causes are less effective, they undermine each other. Taking either one of two desirable jobs leads to happiness for Dave. When taking both, chance of his happiness is reduced due to overstress. For  $C = \{c_1, c_2\}$ , if  $c_1, c_2$  undermine each other, we have  $P(e^1|c_1^1, c_2^0) > 0$ ,  $P(e^1|c_1^0, c_2^1) > 0$ , and

$$P(e^1|c_1^1, c_2^1) < \min(P(e^1|c_1^1, c_2^0), P(e^1|c_1^0, c_2^1)).$$

Def. 1 defines both causal interactions.

**Definition 1** Let  $R = \{W_1, W_2, \dots\}$  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

$$\forall R' P(e^1 \leftarrow \underline{y}^1) \leq P(e^1 \leftarrow \underline{x}^1).$$

Sets of causes in  $R$  undermine each other, iff  $\forall R' P(e^1 \leftarrow \underline{y}^1) > P(e^1 \leftarrow \underline{x}^1)$ .

Reinforcement and undermining occur between individual causes as well as sets of them. When it is between individuals, each  $W_i$  is a singleton. Otherwise,  $W_i$  can be a generic set. Consider  $X = \{c_1, c_2, c_3, c_4\}$ ,  $W_1 = \{c_1, c_2\}$ ,  $W_2 = \{c_3, c_4\}$ ,  $R = \{W_1, W_2\}$ , where  $c_1, c_2$  reinforce each other, and so do  $c_3$  and  $c_4$ . But  $W_1$  and  $W_2$  can undermine each other.

Disjoint sets of causes  $W_1, \dots, W_m$  satisfy *failure conjunction* iff

$$(e^0 \leftarrow \underline{w}_1^1, \dots, \underline{w}_m^1) = (e^0 \leftarrow \underline{w}_1^1) \wedge \dots \wedge (e^0 \leftarrow \underline{w}_m^1).$$

That is, when causes collectively fail to produce the effect, each must have failed to do so. They also satisfy *failure independence* iff

$$\begin{aligned} & P((e^0 \leftarrow \underline{w}_1^1) \wedge \dots \wedge (e^0 \leftarrow \underline{w}_m^1)) \\ &= P(e^0 \leftarrow \underline{w}_1^1) \times \dots \times P(e^0 \leftarrow \underline{w}_m^1). \end{aligned} \quad (3)$$

Disjoint sets of causes  $W_1, \dots, W_m$  satisfy *success conjunction* iff

$$(e^1 \leftarrow \underline{w}_1^1, \dots, \underline{w}_m^1) = (e^1 \leftarrow \underline{w}_1^1) \wedge \dots \wedge (e^1 \leftarrow \underline{w}_m^1).$$

That is, collective success requires individual effectiveness. They also satisfy *success independence* iff

$$\begin{aligned} & P((e^1 \leftarrow \underline{w}_1^1) \wedge \dots \wedge (e^1 \leftarrow \underline{w}_m^1)) \\ &= P(e^1 \leftarrow \underline{w}_1^1) \times \dots \times P(e^1 \leftarrow \underline{w}_m^1). \end{aligned} \quad (4)$$

Causes are undermining when they satisfy success conjunction and independence. Hence, undermining can be modeled by a direct NIN-AND gate (Fig. 1 (a)). Its root nodes (top) are

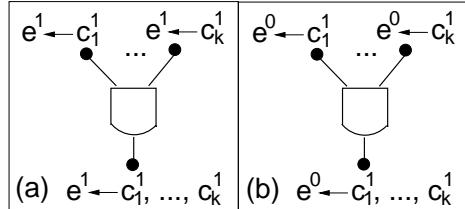


Figure 1: (a) A direct NIN-AND gate. (b) A dual NIN-AND gate.

single-causal successes, and its leaf (bottom) is the multi-causal success in question. Success conjunction is expressed by NIN-AND gate, and success independence by disconnection of roots other than through the gate. Probability of leaf event is computed by Eqn. (4). Similarly, causes are reinforcing when they satisfy failure conjunction and independence. Hence, reinforcement can be modeled by a dual gate (Fig. 1 (b)). Leaf event probability is obtained by Eqn. (3).

By organizing multiple direct and dual NIN-AND gates in a tree, mixture of reinforcement and undermining at multiple levels can be expressed in an NIN-AND tree model. Consider  $C = \{c_1, c_2, c_3\}$ , where  $c_1$  and  $c_3$  undermine each other, but collectively they reinforce  $c_2$ . Assuming event conjunction and independence, their interaction relative to event  $e^1 \leftarrow c_1^1, c_2^1, c_3^1$  can be expressed by NIN-AND tree in Fig. 2. Top gate is direct and bottom gate (leaf gate) is

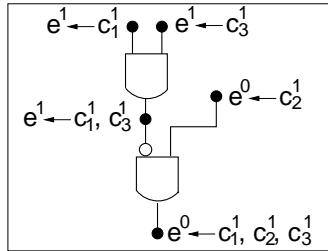


Figure 2: An NIN-AND tree

dual. Link downward from node  $e^1 \leftarrow c_1^1, c_3^1$  has a white oval end (a negation link) and negates the event into  $e^0 \leftarrow c_1^1, c_3^1$ . All other links are forward links. Probability of leaf event is computed by Eqns. (3) and (4). For instance, from single-causal probabilities for root events,  $P(e^1 \leftarrow c_1^1) = 0.85$ ,  $P(e^1 \leftarrow c_2^1) = 0.8$ ,  $P(e^1 \leftarrow c_3^1) = 0.7$ , derive  $P(e^0 \leftarrow c_1^1, c_2^1, c_3^1)$  as follows:

$$\begin{aligned} P(e^1 \leftarrow c_1^1, c_3^1) &= 0.85 \times 0.7 = 0.595 \\ P(e^0 \leftarrow c_1^1, c_2^1, c_3^1) &= P(e^0 \leftarrow c_1^1, c_3^1)P(e^0 \leftarrow c_2^1) \\ &= 0.405 \times 0.2 = 0.081 \end{aligned}$$

Hence,  $P(e^1 \leftarrow c_1^1, c_2^1, c_3^1) = 0.919$  by Eqn. (2).

An NIN-AND tree is *minimal* if consecutive gates are opposite in type, e.g., Fig. 2. Every NIN-AND tree model is equivalent to a unique minimal NIN-AND tree model (Xiang et al., 2009). In this work, we assume minimal models.

### 3 MF of Direct Gate Local Models

Consider the BN family in Fig. 3 (a), where causal interaction is modeled by direct NIN-AND gate in Fig. 1 (a) with  $k = n$  and with single-causal probabilities  $P(e^1 \leftarrow c_i^1)$  ( $i = 1, \dots, n$ ) specified.

A *MF* of a direct NIN-AND gate model consists of a Markov network (MN) (Pearl, 1988)

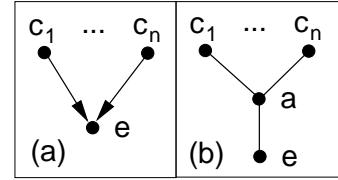


Figure 3: (a) A BN family. (b) A MN segment.

segment and a set of generalized potentials. The MN segment contains the BN family variables plus auxiliary variable  $a$  of domain  $\{a^0, a^1, a^2\}$  or simply  $\{0, 1, 2\}$ . They are connected into an *undirected* graph as Fig. 3 (b).

For each link  $\langle x, y \rangle$  in the MN segment, a *generalized potential* made of reals, possibly negative, is assigned. Table 1 shows potentials  $f(x, y)$  for links  $\langle c_i, a \rangle$  and  $\langle a, e \rangle$ . Let

Table 1:  $f(a, c_i)$  (left) and  $f(e, a)$  (right) of a direct NIN-AND gate model

| $a$ | $c_i$ | $f(a, c_i)$               | $e$ | $a$ | $f(e, a)$ |
|-----|-------|---------------------------|-----|-----|-----------|
| 0   | 0     | 1                         | 0   | 0   | -1        |
| 0   | 1     | $P(e^1 \leftarrow c_i^1)$ | 0   | 1   | 1         |
| 1   | 0     | 1                         | 0   | 2   | 1         |
| 1   | 1     | 1                         | 1   | 0   | 1         |
| 2   | 0     | 1                         | 1   | 1   | 0         |
| 2   | 1     | 0                         | 1   | 2   | -1        |

$g(e, c_1, \dots, c_n)$  denote product of link potentials with variable  $a$  marginalized out,

$$g(e, c_1, \dots, c_n) = \sum_a f(e, a) \prod_{i=1}^n f(a, c_i).$$

We refer to  $g(e, c_1, \dots, c_n)$  as *marginalized product*. Theorem 1 shows  $g(e, c_1, \dots, c_n)$  to be the CPT defined by the direct NIN-AND gate model.

**Theorem 1** Given a MF of a direct NIN-AND gate model over  $e$  and  $C = \{c_1, \dots, c_n\}$ , then the marginalized product  $g(e, c_1, \dots, c_n)$  is the CPT of the direct gate model, i.e.,  $g(e, c_1, \dots, c_n) = P(e | c_1, \dots, c_n)$ .

Proof: Start from Fig. 4, and then consider Eqn. (5) for  $e = e^1$ ,

$$\prod_{i=1}^n f(a^0, c_i) - \prod_{i=1}^n f(a^2, c_i).$$

$$\begin{aligned}
g(e, c_1, \dots, c_n) &= f(e, a^0) \prod_{i=1}^n f(a^0, c_i) + f(e, a^1) \prod_{i=1}^n f(a^1, c_i) + f(e, a^2) \prod_{i=1}^n f(a^2, c_i) \\
&= \begin{cases} f(e^0, a^0) \prod_{i=1}^n f(a^0, c_i) + f(e^0, a^1) \prod_{i=1}^n f(a^1, c_i) + f(e^0, a^2) \prod_{i=1}^n f(a^2, c_i) & \text{if } e = e^0 \\ f(e^1, a^0) \prod_{i=1}^n f(a^0, c_i) + f(e^1, a^1) \prod_{i=1}^n f(a^1, c_i) + f(e^1, a^2) \prod_{i=1}^n f(a^2, c_i) & \text{if } e = e^1 \end{cases} \\
&= \begin{cases} -\prod_{i=1}^n f(a^0, c_i) + \prod_{i=1}^n f(a^1, c_i) + \prod_{i=1}^n f(a^2, c_i) & \text{if } e = e^0 \\ \prod_{i=1}^n f(a^0, c_i) - \prod_{i=1}^n f(a^2, c_i) & \text{if } e = e^1 \end{cases} \\
&= \begin{cases} 1 - (\prod_{i=1}^n f(a^0, c_i) - \prod_{i=1}^n f(a^2, c_i)) & \text{if } e = e^0 \\ \prod_{i=1}^n f(a^0, c_i) - \prod_{i=1}^n f(a^2, c_i) & \text{if } e = e^1 \end{cases} \tag{5}
\end{aligned}$$

Figure 4: Proof of Prop. 1 (see Table 1 for definition of factors)

If for one or more  $i$ ,  $c_i = c_i^1$ , it becomes  $\prod_{c_i=c_i^1} P(e^1 \leftarrow c_i^1)$ , and satisfies  $P(e^1 \leftarrow c_1, \dots, c_n)$  from Eqn. (4). If for every  $i$ ,  $c_i = c_i^0$ , it becomes  $1 - 1 = 0$ , and satisfies Eqn. (1). Therefore,  $g(e^1, c_1, \dots, c_n)$  corresponds to  $P(e^1 | c_1, \dots, c_n)$  as defined by direct gate model.

From Eqn. (5) for  $e = e^0$ , it follows that  $g(e, c_1, \dots, c_n)$  defines the CPT  $P(e | c_1, \dots, c_n)$  (cf. Eqn. (2)) and the theorem follows.  $\square$

#### 4 MF of Dual Gate Local Models

Next, consider the BN family in Fig. 3 (a), modeled by the dual NIN-AND gate in Fig. 1 (b) with  $k = n$  and with single-causal probabilities specified.

A *MF of a dual NIN-AND gate model* consists of a MN segment (Fig. 3 (b)), and a set of generalized potentials, one per link. Table 2 shows potentials for links  $\langle c_i, a \rangle$  and  $\langle a, e \rangle$ .

Table 2:  $f(a, c_i)$  (left) and  $f(e, a)$  (right) of a dual NIN-AND gate model

| $a$ | $c_i$ | $f(a, c_i)$               | $e$ | $a$ | $f(e, a)$ |
|-----|-------|---------------------------|-----|-----|-----------|
| 0   | 0     | 1                         | 0   | 0   | 1         |
| 0   | 1     | $P(e^0 \leftarrow c_i^1)$ | 0   | 1   | 0         |
| 1   | 0     | 1                         | 0   | 2   | 0         |
| 1   | 1     | 1                         | 1   | 0   | -1        |
| 2   | 0     | 1                         | 1   | 1   | 1         |
| 2   | 1     | 0                         | 1   | 2   | 0         |

Theorem 2 shows that the marginalized product of link potentials is the CPT defined by the dual NIN-AND gate model. Its proof is omitted due to space.

**Theorem 2** *Given a MF of a dual NIN-AND gate model over  $e$  and  $C = \{c_1, \dots, c_n\}$ , the marginalized product is the CPT of dual gate model, i.e.,  $g(e, c_1, \dots, c_n) = P(e | c_1, \dots, c_n)$ .*

#### 5 MF of Tree Local Models

Secs. 3 and 4 deal with MFs of single-gate NIN-AND tree models. We now consider general tree models. Let a BN family over  $e$  and  $C = \{c_1, \dots, c_n\}$  be modeled by an NIN-AND tree  $T$  and single-causal probabilities.

A *MF of an NIN-AND tree model* consists of a MN segment and a set of generalized potentials, one per link. The MN segment  $G$  is obtained from  $T$  as follows:

1. For each root in  $T$ , labeled  $e \leftarrow c_i^1$ , relabel it by the cause variable  $c_i$ .
2. For each gate, relabel its output node with an auxiliary variable,  $a \in \{a^0, a^1, a^2\}$ , connect its input nodes to  $a$ , and delete the gate. If the gate deleted is direct or dual, refer to variable  $a$  as direct or dual.
3. For leaf gate and corresponding auxiliary variable  $a$ , create a new node, label it by variable  $e$ , and connect  $a$  to  $e$ .

Consider the BN family in Fig. 5 (a), modeled by NIN-AND tree in Fig. 2. The MN segment is shown in Fig. 5 (b), where  $a, b$  are auxiliary.

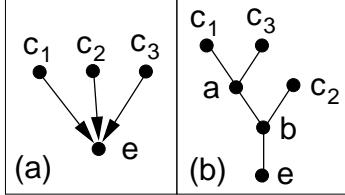


Figure 5: (a) A BN family. (b) A MN segment.

Link potentials are assigned as follows, where  $a, b$  are auxiliary variables.

1. For each link  $\langle c_i, a \rangle$ , where  $a$  is direct, assign the potential as Table 1 (left).
2. For each link  $\langle c_i, a \rangle$ , where  $a$  is dual, assign the potential as Table 2 (left).
3. For each link  $\langle a, b \rangle$ , where  $b$  is closer to  $e$  than  $a$ , assign the potential as Table 3.

Table 3:  $f(a, b)$  for an NIN-AND tree model

| $b$ | $a$ | $f(a, b)$ | $b$ | $a$ | $f()$ | $b$ | $a$ | $f()$ |
|-----|-----|-----------|-----|-----|-------|-----|-----|-------|
| 0   | 0   | -1        | 1   | 0   | 0     | 2   | 0   | 0     |
| 0   | 1   | 1         | 1   | 1   | 1     | 2   | 1   | 0     |
| 0   | 2   | 1         | 1   | 2   | 0     | 2   | 2   | 1     |

4. For each link  $\langle a, e \rangle$ , where  $a$  is direct, assign the potential as Table 1 (right).
5. For each link  $\langle a, e \rangle$ , where  $a$  is dual, assign the potential as Table 2 (right).

We denote an NIN-AND tree model as  $TM = (e, C, T, SP)$ , where  $T$  is the NIN-AND tree and  $SP$  is the set of single-causal probabilities one per cause in  $C$ . We denote the MF of  $TM$  as  $\phi = (e, C, G, F)$ , where  $G$  is the MN segment and  $F$  is the set of potentials over links of  $G$ . Prop. 1 characterizes the MN segment, whose proof is straightforward.

**Proposition 1** Let  $TM = (e, C, T, SP)$  be an NIN-AND tree model, where  $T$  contains  $m \geq 1$  gates. Let  $\phi = (e, C, G, F)$  be the MF of  $TM$ . The following hold for  $G$ :

1.  $G$  is a tree of  $m$  internal nodes, all of which are auxiliary variables, one per gate of  $T$ .
2. Terminal nodes (degree = 1) are  $c_1, \dots, c_n$  and  $e$  only.

Although  $G$  is undirected, we refer to groups of nodes through their causal relation. For each internal node, nodes along causal direction to  $e$  are *downstream* and nodes against the direction to a  $c_i$  are *upstream*. Links of  $G$  and link potentials are referred to accordingly as upstream or downstream. In Fig. 5 (b), node  $c_1$ , link  $\langle c_1, a \rangle$ , and potential  $f(a, c_1)$  are upstream to  $a$ , while  $b$  is downstream to  $a$ .

## 6 Soundness of MF of Tree Models

We show that  $\phi$  represents the CPT of  $TM$  exactly. As it has been shown in Theorems 1 and 2 for single-gate  $TM$ s, we focus on multi-gate  $TM$ s below. First, we define a condition on potentials of  $\phi$  in a subtree rooted at a link  $\langle x, y \rangle$ .

**Definition 2** Let  $\phi = (e, C, G, F)$  be the MF of an NIN-AND tree model  $TM = (e, C, T, SP)$  of two or more gates. Let  $y$  be an internal node of  $G$ ,  $x$  be an upstream neighbor of  $y$ ,  $c_1, \dots, c_k$  ( $k \geq 1$ ) be upstream causes of  $y$  via link  $\langle x, y \rangle$ , and  $R$  be the set of internal nodes upstream of  $y$  via link  $\langle x, y \rangle$ . Let  $g(b, R, c_1, \dots, c_k)$  be the product of potentials assigned to links upstream of  $y$ . Then

$$g(y, c_1, \dots, c_k) = \sum_R g(y, R, c_1, \dots, c_k)$$

is a **valid marginalized product** (VMP) iff the following hold:

$$g(y^0, c_1^0, \dots, c_k^0) = 1; \quad (6)$$

$$g(y^0, c_1, \dots, c_k) = P(e \leftarrow c_u^1, \dots, c_v^1), \quad (7)$$

where one or more  $c_i = c_i^1$ , denoted  $c_u, \dots, c_v$ , and  $e = e^1$  ( $e^0$ ) if  $y$  is direct (dual);

$$g(y^1, c_1, \dots, c_k) = 1; \quad (8)$$

$$g(y^2, c_1^0, \dots, c_k^0) = 1; \quad (9)$$

$$g(y^2, c_1, \dots, c_k) = 0, \quad (10)$$

where one or more  $c_i = c_i^1$ .

Lemma 1 shows that the VMP condition holds when  $x$  is a root.

**Lemma 1** *If  $x$  is a cause  $c_i$ , then  $g(y, c_i)$  is a VMP.*

Proof: Here,  $R = \emptyset$ ,  $k = 1$ , and  $g(y, c_i) = f(y, c_i)$ . If  $y$  is direct,  $f(y, c_i)$  is as Table 1 (left), If  $y$  is dual,  $f(y, c_i)$  is as Table 2 (left). In either case, Eqns. (6) through (10) hold.  $\square$

Prop. 2 shows that the VMP condition holds for an arbitrary  $y$  node.

**Proposition 2** *For any  $y$ ,  $g(y, c_1, \dots, c_k)$  is a VMP.*

Proof: We prove by induction in the length  $L$  of longest path from  $y$  to an upstream  $c_i$ . Case for  $L = 1$  is shown in Lemma 1. We assume that the statement holds for  $L \leq w$ , where  $w \geq 1$ , and consider  $L = w + 1$ .

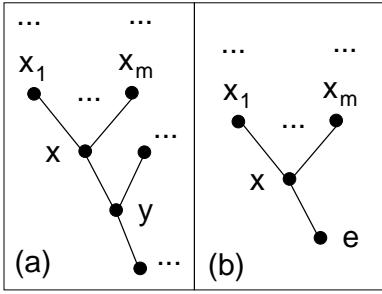


Figure 6: Illustration for proofs of Prop. 2 (a) and Theorem 3 (b)

Suppose  $x$  has upstream neighbors  $x_1, \dots, x_m$  ( $m \geq 2$ ), (see Fig. 6 (a)). Denote corresponding upstream cause sets as  $S_1, \dots, S_m$  that form a partition of  $c_1, \dots, c_k$ . Denote marginalized product of  $f(x, x_i)$  and potentials upstream of  $x_i$  by  $g_i(x, S_i)$ , where  $i = 1, \dots, m$ . By inductive assumption, each  $g_i(x, S_i)$  is a VMP. We have

$$g(y, c_1, \dots, c_k) = \sum_x f(y, x) \prod_{i=1}^m g_i(x, S_i).$$

From Table 3 for  $f(y, x)$ , we have

$$g(y, c_1, \dots, c_k)$$

$$= \begin{cases} -\prod_{i=1}^m g_i(x^0, S_i) \\ +\prod_{i=1}^m g_i(x^1, S_i) \\ +\prod_{i=1}^m g_i(x^2, S_i) & (y = y^0) \\ \prod_{i=1}^m g_i(x^1, S_i) & (y = y^1) \\ \prod_{i=1}^m g_i(x^2, S_i) & (y = y^2). \end{cases}$$

From Eqns. (6), (8), (9) for  $g_i(x, S_i)$ , we have

$$-\prod_{i=1}^m g_i(x^0, \underline{s}_i^0) + \prod_{i=1}^m g_i(x^1, \underline{s}_i^0) + \prod_{i=1}^m g_i(x^2, \underline{s}_i^0) \\ = -1 + 1 + 1.$$

Hence, Eqn. (6) holds for  $g(y, c_1, \dots, c_k)$ .

Next, consider Eqn. (7). Since  $T$  is minimal, if  $y$  is direct,  $x$  is dual. From Eqns. (6), (7) for  $g_i(x, S_i)$ ,

$$-\prod_{i=1}^m g_i(x^0, S_i) = -\prod_{c_i \in S_i, c_i = c_i^1} P(e^0 \leftarrow c_i^1).$$

From Eqn. (8),  $\prod_{i=1}^m g_i(x^1, S_i) = 1$ . From Eqn. (10),  $\prod_{i=1}^m g_i(x^2, S_i) = 0$ . Hence, we have

$$-\prod_{i=1}^m g_i(x^0, S_i) + \prod_{i=1}^m g_i(x^1, S_i) + \prod_{i=1}^m g_i(x^2, S_i) \\ = 1 - \prod_{c_i \in S_i, c_i = c_i^1} P(e^0 \leftarrow c_i^1) = P(e^1 \leftarrow c_u^1, \dots, c_v^1),$$

by Eqn. (3), where  $c_u, \dots, c_v$  are those in  $c_1, \dots, c_k$  with  $c_i = c_i^1$ .

Similarly, if  $y$  is dual, we have

$$g(y^0, c_1, \dots, c_k) = P(e^0 \leftarrow c_u^1, \dots, c_v^1).$$

Hence, Eqn. (7) holds for  $g(y, c_1, \dots, c_k)$ .

From Eqn. (8) for  $g_i(x, S_i)$ , we have Eqn. (8) for  $g(y, c_1, \dots, c_k)$ . From Eqn. (9) for  $g_i(x, S_i)$ , it holds for  $g(y, c_1, \dots, c_k)$ . Finally, from Eqn. (10) for  $g_i(x, S_i)$ , we have Eqn. (10) for  $g(y, c_1, \dots, c_k)$ .  $\square$

Theorem 3 concludes soundness of  $\phi$ .

**Theorem 3** *Let  $\phi = (e, C, G, F)$  be the MF of an NIN-AND tree model  $TM = (e, C, T, SP)$  of two or more gates. Then the marginalized product of all potentials satisfies  $g(e, c_1, \dots, c_n) = P(e|c_1, \dots, c_n)$ .*

Proof sketch: Due to space limit, a proof sketch is given below. Let  $x$  be the (only) neighbor of  $e$  in  $G$ , shown in Fig. 6 (b), upstream neighbors of  $x$  be  $x_1, \dots, x_m$  ( $m \geq 2$ ), and corresponding upstream cause sets be  $S_1, \dots, S_m$  that partition  $c_1, \dots, c_n$ . Denote marginalized product of  $f(x, x_i)$  and potentials upstream of  $x_i$  by  $g_i(x, S_i)$ , where  $i = 1, \dots, m$ .

Consider the case where  $x$  is direct. From Table 1, Prop. 2, and Eqn. (8), it can be shown that

$$g(e^0, c_1, \dots, c_n) = 1 - g(e^1, c_1, \dots, c_n).$$

Applying Eqns. (1), (4), (6), (7), (9) and (10) to  $g(e^1, c_1, \dots, c_n)$ , it can be shown that  $g(e, c_1, \dots, c_n)$  corresponds to  $P(e|c_1, \dots, c_n)$  defined by TM.

The case for dual  $x$  can be similarly shown, and the theorem follows.  $\square$

## 7 Compiling NIN-AND Tree Modeled BNs for LP

Consider a binary BN over variable set  $V$  with DAG  $D$ . Each root of  $D$  is assigned a prior, collected in set  $PR$ . Each single-parent non-root is assigned a CPT, collected in set  $PS$ . Family of each multi-parent non-root is expressed as an NIN-AND tree model, collected in set  $\Psi$ . Then,  $\Gamma = (V, D, PR, PS, \Psi)$  is an *NIN-AND tree modeled BN* (NATBN).

For effective inference, we first convert  $\Gamma$  into a Markov network as follows:

1. Each root  $x$  of  $D$  retains prior  $P(x) \in PR$ .
2. For each non-root  $x$  with a single parent  $y$ , drop direction in link  $y \rightarrow x$ , and assign  $P(x|y) \in PS$  to link  $\langle y, x \rangle$ .
3. For each multi-parent non-root  $x$  in  $D$  with parent set  $\pi(x)$ , and the NIN-AND tree model  $TM \in \Psi$  over family  $\{x\} \cup \pi(x)$ , derive the MF  $\phi$  of  $TM$ .

Replace subgraph of  $D$  over  $\{x\} \cup \pi(x)$  (e.g., Fig. 5 (a)) by the MN segment of  $\phi$  (e.g., Fig. 5 (b)). Assign each link of the MN segment a potential according to  $\phi$ .

We denote the result  $\Delta = (V^+, D^+, PR, LT)$ , where  $V^+$  is the set  $V$  of variables plus all auxiliary variables,  $D^+$  is the resultant undirected graph over node set  $V^+$ ,  $PR$  is the set of potentials associated with nodes in a subset of  $V^+$ , and  $LT$  is the set of link potentials. We refer to  $\Delta$  as the *multiplicatively factorized MN* (MFMN) of  $\Gamma$ .

Next, we compile  $\Delta$  for LP as follows:

1. Triangulate  $D^+$  and construct a JT  $T^+$ .
2. For each potential in  $PR$  and  $LT$ , assign to a cluster in  $T^+$  that contains corresponding variables.

$T^+$  can then be used for LP. The above does not involve moralization. Note that each potential in  $T^+$  is over at most two variables.

## 8 Efficiency Improvement

A collection of 120 NATBNs are simulated, divided into 4 sets of 30 each. Each NATBN contains 100 variables. For NATBNs in the same set, the number of causes per NIN-AND tree model, i.e.,  $n$ , is identically upper-bounded. The bounds are 5, 7, 9 and 11, respectively. All NATBNs have the same density (5% more links than singly-connected). Each is compiled into a MFMN for LP.

For comparison, from each NATBN, a *peer BN* is derived by assigning each multi-parent variable the CPT determined from its NIN-AND tree model. Peer BNs are compiled for LP normally.

For each NATBN and its peer BN, random observations over 5 variables are used in inference by LP. Table 4 summarizes results. Each row, indexed by  $n$ , contains results from one set of NATBNs. Each column shows sample mean (left) and standard deviation (right) of a given measure. Col. 2,  $pjts$ , indicates state space sizes of JTs compiled from peer BNs. Col. 3,  $mjts$ , shows state space sizes of JTs from NATBNs. Last two columns,  $pinf$  and  $minf$ , show LP time to compute posterior marginals for all variables using peer BNs and NATBNs.

As  $n$  grows, JT state space sizes for peer BNs grow rapidly, but only slightly for NATBNs.

Table 4: Experimental Results

| n  | pjts   |       | mjts   |       | pinf  | (ms)  | minf | (ms) |
|----|--------|-------|--------|-------|-------|-------|------|------|
| 5  | 1232.8 | 78.4  | 1796.8 | 150.6 | 53.0  | 28.7  | 41.1 | 11.2 |
| 7  | 1810.0 | 164.5 | 1899.8 | 189.0 | 68.0  | 13.5  | 44.9 | 15.0 |
| 9  | 2784.5 | 444.9 | 1916.5 | 205.6 | 146.8 | 43.4  | 53.3 | 18.4 |
| 11 | 5919.3 | 815.8 | 1846.5 | 166.0 | 566.4 | 269.4 | 49.9 | 20.0 |

Accordingly, computational savings of NATBNs grows with  $n$ . When  $n = 5$ , NATBNs use about 80% of inference time as peer BNs. When  $n = 11$ , inference time of NATBNs is one-order of magnitude less than peer BNs.

## 9 Conclusion

This contribution proposes a multiplicative factorization of NIN-AND tree modeled BNs and compilation of the resultant graphical model for LP. We have shown that the method allows exact inference for posterior marginals of all variables. Our experiments demonstrate a significant efficiency gain for sparse BNs with large family sizes. Research extending the method to multi-valued variables is ongoing.

## Acknowledgments

I thank reviewers for their helpful comments, and apologize for not being able to address all of them fully due to space limit. Financial support from NSERC, Canada is acknowledged.

## 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.
- D. Heckerman. 1993. Causal independence for knowledge acquisition and inference. In D. Heckerman and A. Mamdani, editors, *Proc. 9th Conf. on Uncertainty in Artificial Intelligence*, pages 122–127. Morgan Kaufmann.
- F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. 1990. Bayesian updating in causal probabilistic net- works by local computations. *Computational Statistics Quarterly*, (4):269–282.
- 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.
- A.L. Madsen and B. D’Ambrosio. 2000. A factorized representation of independence of causal influence and lazy propagation. *Inter. J. Uncertainty, Fuzziness and Knowledge-Based Systems*, 8(2):151–166.
- K.G. Olesen, U. Kjærulff, F. Jensen, F.V. Jensen, B. Falck, S. Andreassen, and S.K. Andersen. 1989. A munin network for the median nerve—a case study on loops. *Applied Artificial Intelligence*, 3(2-3):385–403.
- J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- P. Savicky and J. Vomlel. 2007. Exploiting tensor rank-one decomposition in probabilistic inference. *Kybernetika*, 43(5):747–764.
- M. Takikawa and B. D’Ambrosio. 1999. Multiplicative factorization of noisy-max. In *Proc. 15th Conf. Uncertainty in Artificial Intelligence*, pages 622–630.
- 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. 2009. Towards effective elicitation of NIN-AND tree causal models. In L. Godo and A. Pugliese, editors, *Inter. Conf. on Scalable Uncertainty Management (SUM 2009)*, LNCS 5785, pages 282–296. Springer-Verlag Berlin Heidelberg.
- N. Zhang and D. Poole. 1996. Exploiting causal independence in bayesian network inference. *J. Artificial Intelligence Research*, 5:301–328.

## Subject Index

- B-spline interpolation, 211
- Bayesian model averaging, 91, 147
- Bayesian networks, 11, 27, 51, 59, 131, 195, 227, 235, 243, 259, 267, 291, 331, 339, 355, 363
- Bayesian networks classifiers, 19
- Bayesian odds ratio, 147
- Bayesian Dirichlet equivalence, 331
- CPT decomposition, 355
- Chow-Liu algorithm, 315
- Dirichlet score, 331
- Gaussian networks, 19
- Gaussian process, 115
- Gibbs sampling, 91
- LP relaxation, 307
- MAP estimation, 75
- MDL Markov networks, 315
- MDPs, 11
- Markov equivalence, 123
- Markov trees, 283
- PC algorithm, 251
- PERT networks, 275
- POMDPs, 11
- XML, 11
- active learning, 3, 123
- application to agriculture, 259
- application to flood damage, 347
- application to medicine, 75, 219
- applications, 75, 219, 259, 347
- approximate computation, 43
- asymmetric decision problems, 83
- belief update, 227
- bias-variance trade-off, 283
- bidirected graph, 299
- binary classifiers, 139
- binary hidden manifest, 291
- bioinformatics, 19
- bounded error, 235
- canonical models, 355
- categorical data, 203
- causal independence models, 363
- causal inference, 123
- causality, 35
- censoring, 115
- chain classifiers, 139
- chain graph, 179, 251, 299
- characteristic imset, 307
- classification, 75, 91
- co-variation alternatives, 267
- computational complexity, 187
- computational complexity of reasoning, 51
- context trees, 91
- copulas, 155
- data cleaning methods, 243
- data complexity measures, 107
- data mining, 347
- decision analysis, 83
- decision-theoretic troubleshooting, 195
- density estimation, 155
- deterministic algorithms, 43
- deterministic variables, 59
- diagnostics, 291
- discretization, 347
- disease modeling, 179
- dynamic programming, 91
- dynamic Bayesian networks, 219
- effect size, 147
- ensemble methods, 283
- essential graph, 307
- exact inference, 171
- expectation maximization, 75
- expectation propagation, 115
- expert interaction, 243
- factor algebra, 99
- gene regulatory network, 339
- graphical models, 35, 123
- hardness of approximation, 195
- heuristic search, 235
- high-dimension, 323
- hybrid domains, 163
- hybrid Bayesian networks, 171, 211, 275
- imprecise probability, 3
- indeterminacy and granularities, 131
- inequalities constraints, 291
- inference, 99, 275, 355

- influence diagrams, 11, 51, 83  
influence diagrams inference, 43  
interactive learning, 27  
interventions, 123  
  
latent tree models, 203  
latent variables, 91, 155  
lazy propagation, 227, 363  
learning, 163, 251, 299  
learning Bayesian networks, 27, 307  
local search, 339  
log-normal, 67  
  
machine learning, 259, 331  
mass spectra classification, 19  
maximal ancestral graphs, 35  
maximum likelihood, 163  
missing data, 75, 323  
mixed graphical Markov models, 323  
mixture models, 91, 283  
mixtures of polynomials, 59, 211, 275  
mixtures of truncated basis functions, 163, 171  
mixtures of truncated exponentials, 59, 67, 275  
model-based clustering, 203  
multi-label classification, 107, 139  
multidimensional clustering, 203  
multivariate regression chain graph, 299  
  
naive credal classifier, 3  
naive Bayes, 3, 107, 211  
noisy evidence, 243  
noisy-threshold models, 355  
non-impeding noisy-AND tree models, 363  
non-parametric density estimation, 211  
  
parsimonious Markov models, 91  
phylogenetic trees, 291  
prediction, 259  
probabilistic graphical model, 11  
probabilistic inference, 187, 355, 363  
probability trees, 43  
  
qualitative abstraction, 179  
  
re-approximation, 275  
robust decision making, 51  
  
semi-naive Bayesian network classifiers, 107  
sensitivity analysis, 267  
software framework, 99

## Author Index

- Antal, Peter, 147  
 Antonucci, Alessandro, 3  
 Arias, Manuel, 11, 27  
 Bühlmann, Peter, 123  
 Bellón, Victor, 19  
 Bermejo, Íñigo, 11, 27  
 Bielza, Concha, 211  
 Borboudakis, Giorgos, 35  
 Bourguignon, Pierre-Yves, 91  
 Butz, Cory J., 227  
 Cano, Andrés, 43  
 Castelo, Robert, 323  
 Cerquides, Jesús, 19  
 Chen, Suming, 51  
 Choi, Arthur, 51  
 Cobb, Barry R., 59, 67  
 Corani, Giorgio, 3, 75  
 Díez, Francisco Javier, 11, 27, 83  
 Darwiche, Adnan, 51  
 Druždžel, Marek J., 219  
 Eggeling, Ralf, 91  
 Evers, Sander, 99  
 Flores, M. Julia, 107, 259  
 Gámez, José A., 107  
 Gómez-Olmedo, Manuel, 43  
 Gabaglio, Sandra, 3  
 Gambardella, Luca, 75  
 Geurts, Pierre, 283  
 Gianaroli, Luca, 75  
 Giusti, Alessandro, 75  
 Givry, Simon de, 339  
 Gohr, André, 91  
 Groot, Perry, 115  
 Grosse, Ivo, 19, 91  
 Hauser, Alain, 123  
 Heijden, Maarten van der, 131  
 Hernandez-Leal, Pablo, 139  
 Hommersom, Arjen, 179  
 Hullam, Gabor, 147  
 König, Caroline Leonore, 83  
 Kirshner, Sergey, 155  
 Kreibich, Heidi, 347  
 López-Cruz, Pedro L., 211  
 Lín, Václav, 195  
 Langseth, Helge, 163, 171  
 Lappenschaar, Martijn, 179  
 Larrañaga, Pedro, 211  
 Li, Chao, 187  
 Liu, Hua, 203  
 Liu, Tengfei, 203  
 Lucas, Peter J.F., 99, 115, 131, 179  
 Lopińska-Dubicka, Anna, 219  
 Luque, Manuel, 83  
 Madsen, Anders L., 227  
 Magli, Cristina, 75  
 Malone, Brandon, 235  
 Mangin, Brigitte, 339  
 Martínez, Ana M., 107  
 Masegosa, Andrés R., 243  
 Merz, Bruno, 347  
 Moral, Serafín, 243  
 Morales, Eduardo F., 139  
 Nicholson, Ann E., 259  
 Nielsen, Thomas Dhyre, 163, 171  
 Oliva, Jesús, 27  
 Orihuela-Espina, Felipe, 139  
 Pérez-Ariza, Cora B., 259  
 Palacios-Alonso, Miguel, 11  
 Paz, Rafael Cabañas de, 43  
 Peña, Jose M., 251, 299  
 Poon, Kin Man, 203  
 Renooij, Silja, 267  
 Riggelsen, Carsten, 347  
 Rumí, Rafael, 67, 171, 275  
 Salmerón, Antonio, 67, 163, 171, 275  
 Scherbaum, Frank, 347  
 Schnitzler, François, 283  
 Shenoy, Prakash P., 59, 275  
 Shiers, Nathaniel, 291  
 Smith, Jim Q., 291

Sonntag, Dag, 299  
Studený, Milan, 307  
Sucar, L. Enrique, 139  
Suzuki, Joe, 315

Tichavský, Petr, 355  
Triantafillou, Sofia, 35  
Tsamardinos, Ioannis, 35  
Tur, Inma, 323

Ueno, Maomi, 187, 331  
Uto, Masaki, 331

Vandel, Jimmy, 339  
Vogel, Kristin, 347  
Vomlel, Jiří, 355

Wang, Yi, 203  
Wehenkel, Louis, 283

Xiang, Yang, 363

Yuan, Changhe, 235

Zhang, Nevin L., 203



# دیزاین ۲۰۱۲



Sponsored by



ugr

Universidad  
de Granada



Escuela Téc. Superior  
de Ing. Informática  
y de Telecomunicación



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE ECONOMÍA  
Y COMPETITIVIDAD

D<sup>2</sup>EZIDE

  
**DECSAI**  
Departamento de Ciencias  
de la Computación e I.A.  
Universidad de Granada



**HUGINEXPERT**  
The leading decision support tool

  
**CITIC-UGR**